NTNU – Trondheim
Norwegian University of
Science and Technology

Fleet Deployment Optimization in Liner Shipping

Sondre Morten Steen

Industrial Economics and Technology Management
Submission date: June 2012
Supervisor: Lars Magnus Hvattum, IØT
Co-supervisor: Inge Norstad, IØT

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

Abstract

This thesis presents a study of a real fleet deployment problem in liner shipping. In the context of data from Saga Forest Carriers and actual contracts between charterers and shipping companies, an optimization model is developed and implemented. The formulated model exploits the advantages of a heterogeneous fleet, incorporates the requirements concerning fairly evenly spread shipments and facilitates overlapping voyages. By additionally including inventory management and freedom of choice regarding speed, the formulation contributes to the existing literature by presenting a rich model featuring new ways to model several issues. The thesis also presents a heuristic solution approach and evaluates the developed model in comparison with the heuristic and well-known alternative solution methods. The computational studies indicate that a rich and realistic mathematical formulation, emphasizing to maintain the existing flexibility, has a positive effect on the results when solving real fleet deployment problems.

Sammendrag

I denne oppgaven studeres ulike problemstillinger knyttet til flåtestyring i linjefart. I lys av virkelige data fra Saga Forest Carriers og kontraktskrav i avtaler mellom charterer og rederi, er en matematisk modell utviklet og implementert. Den formulerte modellen utnytter fordelene av å ha en heterogen flåte, tar hensyn til kravene om jevnt spredte besøk og utnytter overlappende seilaser. Ved å i tillegg inkludere lagerstyring og valgfrihet når det gjelder hastighet, bidrar formuleringen til eksisterende litteratur ved å tilby en rik modell med nye måter for å modellere flere begrensninger. Oppgaven evaluerer også modellens resultater sammenlignet med heuristiske tilnærminger og kjente alternative løsningsmetoder. Resultatene viser at den virkelighetsnære modellen som legger vekt på å ikke kompromittere fleksibilitet, egner seg godt til å løse reelle problemer innen flåtestyring.

Acknowledgements

I would like to express my thanks to my supervisors, Associate Professor Lars Magnus Hvattum from the Department of Industrial Economics and Technology Management (IOT) and Inge Norstad, PhD candidate at IOT and research scientist at MARINTEK for knowledgeable guidance during my work on this thesis.

I am also grateful to Dr. Arvid Steen for his comments on preliminary drafts.

Trondheim, June 5, 2012

Sondre Steen

Contents

1 Introduction **1**
- 1.1 The introduction of operations research . . . 2
- 1.2 Terminology . . . 3
- 1.3 The contribution from this thesis . . . 4

2 Liner shipping **7**
- 2.1 Industrial-, tramp- and liner shipping . . . 7
- 2.2 Planning in liner shipping . . . 7

3 Literature review **9**
- 3.1 Operations research historically . . . 9
- 3.2 The fleet deployment problem . . . 9
  - 3.2.1 Combined inventory management and routing . . . 11
  - 3.2.2 Solution methods . . . 11

4 Problem description **14**

5 Solution methods **17**
- 5.1 Modeling . . . 17
- 5.2 Mathematical formulation . . . 20
- 5.3 Heuristics for large instances . . . 26

6 Computational studies **28**
- 6.1 Input . . . 28
- 6.2 Instances . . . 31
  - 6.2.1 Example instance . . . 32
- 6.3 Testing performance . . . 33
  - 6.3.1 Performance of the full model . . . 33
  - 6.3.2 Fix-and-relax heuristics . . . 35
  - 6.3.3 Problem structure effects . . . 39
- 6.4 Testing the performance relative to other modeling solutions . . . 40
  - 6.4.1 Time windows replacing inventory control . . . 40
  - 6.4.2 Fixed number of voyages . . . 41
  - 6.4.3 Fixed speed . . . 43
  - 6.4.4 Shared inventories . . . 45

7 Concluding remarks **47**

References **49**

A Mathematical formulation **53**

B Mosel code **58**

List of Tables

1 Comparison to Arnulf and Bjørkli [2010]. 5
2 The fleet of Saga Forest Carriers. 29
3 Test instances. 31
4 Example instance. 32
5 Results for the full model (30 and 60 days) 34
6 Results for the full model (120 and 150 days) 35
7 Results for the full model (90 days) 35
8 Specifications for heuristic 1-4. 36
9 Results for heuristic 1 and 2. 37
10 Results for heuristic 3 and 4. 37
11 Results for heuristic 5. 37
12 Results for heuristic 6. 38
13 The best results obtained from the heuristics. 38
14 Voyage distribution between trades. 39
15 Problem structure effects for the full model and when speed is fixed. . . . 39
16 Problem structure effects for time windows and fixed number of voyages. . 39
17 Objective function values when using time windows. 41
18 Solution time and gap when using time windows. 42
19 Objective function values when fixing number of voyages. 43
20 Number of voyages performed. 43
21 The ratio of "low" speed voyages. 44
22 Objective function values when fixing speed. 44
23 Shared inventories. 45
24 Objective function values when sharing inventories. 46

List of Figures

1 The growth of international seaborne trade 1
2 Office locations and trades for Saga Forest Carriers. 14
3 Inventory level as a function of time. 19
4 Overlapping voyages may lead to negative ballast sailing costs. 20
5 Network example for an arc flow model. 21
6 Iteration one in a fix-and-relax heuristic. 27
7 Problem structure for a fix-and-relax heuristic. 27
8 Trades and inventory locations for the example instance. 33
9 Optimal solution for the example instance. 34

1 Introduction

Maritime shipping is a growing transport sector, both in terms of total tonnage and relatively to other means of transport. Globalization of trade and outsourcing of production to low wage countries, mainly in Asia, is the main driver of the growth. Liner shipping is one of the segments that have experienced the largest growth. An annual growth rate of 10% over the past two decades has resulted in an increase in containerized trade from a share of 5.1% of the world's total dry cargo trade in 1980, to 25.4% in 2008 [UNCTAD, 2009].

The economic status in maritime shipping industry follows the worldwide macroeconomic conditions. The development in seaborne trade is tightly connected to the development in the world economy and merchandise trade. Focusing on the last years, UNCTAD [2011] describes an upswing in demand in the world seaborne trade in 2010. The report shows that in 2010, record deliveries of new tonnage were reported with an increase of 28% from 2009, and a 8.6% growth in the world fleet. The container trade segment has had a particular positive trend and since 2005 the containership fleet has nearly tripled. Figure 1 shows the growth in international seaborne trade the last 30 years.

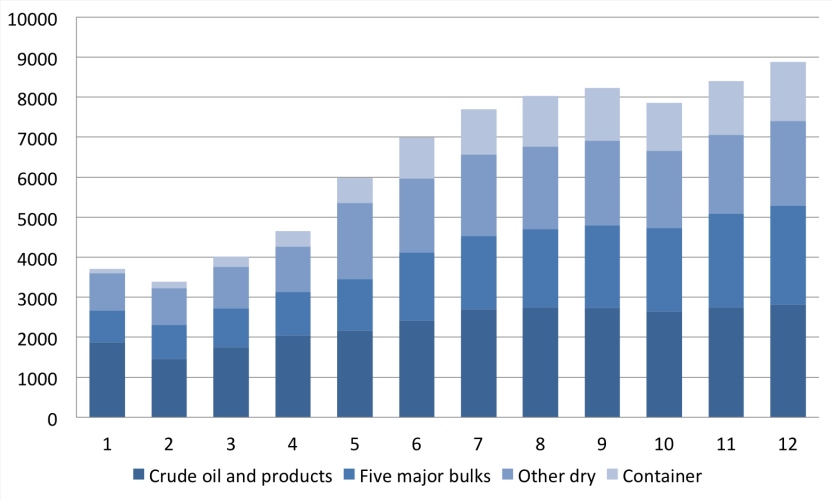

Figure 1: The growth of international seaborne trade (millions of tons) [UNCTAD, 2011].

One explanation of the particularly high increase in transport by sea is the advantages shipping has when it comes to transporting large volumes over long distances at a low cost. Trucks have their advantages when it comes to flexibility and speed, but have limited capacity and range and are relatively costly. While rail has good cargo capacity, transport by rail suffer from lack of freedom of movement and share the same limitations as trucks when it comes to operational area, having to operate on the same continent and in some cases within the same country. Transport by air offer even better flexibility than shipping when it comes to intercontinental transport, but the low capacity and

high cost makes transport by air only a preferred mode for time sensitive, high-value goods.

The challenges in routing and scheduling of ships also distinguish from the challenges of the standard vehicle routing and scheduling problems and the challenges in the airline industry. First, a fleet of ships is much more heterogeneous than fleets of trucks or cargo trains [Ronen, 1983]. Second, trucks often operate out of the same base, while ships do not necessarily have to return to given ports. In scheduling ships there is also much higher uncertainty involved, due to the duration of the voyages and the vulnerability to extreme weather conditions. Long delivery time for new ships and volatile demand also make the economic situation for a shipping company unclear and hard to predict. These differences and more imply that modeling and solving problems in maritime shipping needs to be researched separately from the standard vehicle routing problems. They all support one statement, though: The utilization of the fleet in a competitive and dynamic market is of vital importance.

1.1 The introduction of operations research

Currently the situation in most shipping companies is that the planning process is based on experience, simple spreadsheets and manual modifications. The planners in charge have extensive experience and often first hand knowledge about operating at sea, but in many cases limited education and training when it comes to the use of optimization-based Decision Support Systems (DSSs). Chajakis [1999] describes how the process of introducing computer-based systems among the current staff of planners is a challenging task. Traditionally, the planners have a strong position internally in the company. They might see computer-based systems as a threat to the need for their specialized experience and knowledge. Computer-based DSSs may indeed streamline the planning process, reduce the demand for planners and make the shipping companies less vulnerable to employee turnover, but the knowledge of the experienced planners is still a valuable asset. The computer-based DSSs should be seen as a tool to facilitate a faster planning process and better solution quality.

The environment the shipping companies operate in today is highly competitive. The freight rates are diminishing and new entrants intensify the already fierce competition among the carriers. The lower margins emphasize how crucial it is to operate the service network efficiently and optimize the utilization of the fleet. Furthermore, the fleets are increasing in size, making it undeniable that the planning problems that before were solved manually by the experienced planners, now have become so large and complex that use of computers induce significant savings. At the same time, fuel prices and the economies of scale drive the shipping companies to million dollar investments in increasingly larger containerships. New jumbo ships with capacities around 22 000 TEU (twenty-foot equivalent unit) reduce the shipping price per container with around 40% [Gelareh and Pisinger, 2010], but at the same time, larger ships reinforce the importance of utilization and how even small improvements in the scheduling process induce large

savings. Powell and Perakis [1997] present a case where an integer-programming model was implemented for the fleet deployment at Flota Mercante Grancolombiana. A reduction in operating costs of 1.5% was reported when comparing the integer-programming model to the existing fleet deployment plans.

A sector that has implemented operations research with success is the airline industry. The airline industry faced the same challenges the shipping companies increasingly do today, years ago. Larger fleets, more human resources and a larger network of destinations resulted in overwhelming planning problems. Despite resistance from the organization at first, efficient planning is considered impossible without optimization-based DSSs in the airlines today. As the fleet sizes and the network complexity increase in the shipping industry today, the airline industry has set an example of how optimization-based DSSs are invaluable tools when it comes to achieving economies of scale, cost reductions and improved utilization. Like just-in-time once went from being a competitive advantage to a necessity, these systems are looked upon as a requirement to survive in the airline industry today. The shipping industry still have a way to go, but increasingly seems to recognize the advantages of using optimization-based DSSs when solving highly complex problems.

Lately, the shipping companies seem to recognize the benefit of employing planners with a more academic background [Christiansen et al., 2004]. Their education has equipped them with computer experience, abilities to learn new software fast, and most importantly recognition of the opportunities that exist in new technology. This trend therefore gives reason to hope that the gap between researchers developing optimization-based DSSs and the planners will decrease in the future.

1.2 Terminology

In order to be able to continue the discussion around maritime shipping and operations research, and get more into detail on the liner shipping industry and the fleet deployment problem, it is worthwhile to clarify the terminology used in the rest of the thesis before we proceed.

Liner does in this thesis refer to a cargo ship designated to service scheduled long distance maritime shipping routes, commonly called *trade routes*.

A **trade (route)** describes a number of loading ports in one geographical region and a number of unloading ports in a different, often distant geographical region. In this thesis a trade is referred to by the two regions it services. For example, WCEUR refers to a trade that loads cargo at the US West Coast and unloads in Europe.

A **voyage** is one traversal of a *trade*.

Fleet deployment refers to the decisions regarding which of the available ships to use for each specific *voyage*, on every *trade*.

Charter in is to hire a spot ship from another shipping company to service a *voyage*.

Charter out is to hire out a ship (when excess capacity) to perform services for other shipping companies. A ship can be hired out for a certain amount of time *(time charter)* or on a voyage basis *(voyage charter)*.

In this thesis, **cargo** refers to Saga Forest Carriers' (Saga) most regular cargo, forest products. Saga's fleet are also suited to carry a wide variety of traditional unitised and bulk cargoes, such as containers, steel, clay, aluminium and break-bulk bulk cargoes. The model developed in this thesis is suitable for liners carrying any type of cargo.

The **charterer** is the owner of the transported *cargo* and enter a contract with a liner shipping company for the transportation.

The **Contract of Affreightment (CoA)** is the contract between the ship owner and the *charterer* regarding transport of *cargo*. The most important points are the trade routes, the total volume of *cargo* transported over the planning period, loosely defined time restrictions and the *freight rate*.

The **freight rate** is the price the *charterer* agrees to pay for the transport of the *cargo*.

Ballast sailing is sailing without *cargo*, in other words empty or with only ballast to ensure stability. Ballast sailing is often necessary to reposition a ship for its next *voyage*.

1.3 The contribution from this thesis

This thesis can be seen as an extension of the work of Arnulf and Bjørkli [2010]. Both master theses are written at the Norwegian University of Technology and Science under the supervision of some of the same supervisors. The knowledge of strengths and weaknesses of the work of Arnulf and Bjørkli has been used as an advantage in the process of developing a new, independent model with new features that eliminates some of the limitations of the model from Arnulf and Bjørkli [2010].

Central for the contribution to the current research on the fleet deployment problem of both this thesis and Arnulf and Bjørkli [2010] is the handling of the requirement in the contract between the charterer and the shipping company regarding service times. The CoA often states that a trade should be serviced fairly evenly spread with respect to time. From the shipping company's point of view this is desirable because it provides flexibility when it comes to fleet deployment. Neither do the charterers have any strong incentives to lock the pickup of the goods to specific dates. Arnulf and Bjørkli [2010] state that no studies had previously been performed on this particular aspect of the fleet deployment problem and presented two approaches to the issue. Both were based on restricting the time of service to specified time windows and imposing a minimum threshold for the time between any two voyages on a trade. The first approach used a

Table 1: Comparison to Arnulf and Bjørkli [2010].

Feature	**Arnulf and Bjørkli [2010]**	**This thesis**
Fleet	Homogeneous	Heterogeneous
Number of voyages	Fixed	Optional
Charter in spot ships	Yes	Yes
Accept additional spot cargo	No	Yes
Overlapping voyages	No	Yes
Choice of speed	No	Yes
Inventory control	No	Yes
Quantity control	No	Yes

hard restriction, while the second allowed for the threshold to be broken in return for a penalty addition in the objective function. This thesis uses a hard restriction on the minimum threshold for the time spread between any two voyages on a trade, but uses no limiting time windows for the time of service.

The model in this thesis is also made richer by incorporating a heterogeneous fleet. Most shipping companies have a heterogeneous fleet, and exploiting this opens for different solutions as the choice of type of ship may affect the number of voyages necessary to satisfy the contract requirements.

One limitation in Arnulf and Bjørkli [2010] was that a ship was forced to unload its entire load before accepting any new cargo. This enforced additional costs to situations where the next loading ports for a ship either corresponded to the unloading ports of the current voyage, or could have been visited with a smaller cost than the cost of performing a separate visit after unloading all cargo for the current voyage. This thesis enables the possibility of what we may call overlapping voyages, that is the possibility of loading cargo in available space for the upcoming voyage before the entire current load is discharged.

Offering free choice of speed is a complicating factor that turns the fleet deployment problem into a nonlinear optimization problem. Arnulf and Bjørkli [2010] thus excluded this option. This thesis solved the nonlinearity issue by offering the planners to choose operational speed from a discrete set when assigning a ship to a voyage. This offers the possibility of reducing cost by traveling at an economic and fuel efficient speed when time is available, or to choose a high speed to get the possibility of using a ship on a voyage it would have been unavailable for otherwise.

This thesis also offer a richer portfolio of chartering out options. Arnulf and Bjørkli [2010] subtract an income fee from the total costs whenever a ship has a period of time above a threshold at which the ship do not service voyages. This thesis offers spot voyages equivalent to the contractual voyages, except for the obligation the shipping company has to fulfill them. This thesis also offer the possibility to accept up to a

specified amount of additional spot cargo on a voyage if excess capacity exists, which is a new feature. Finally, the quantity loaded on each ship at every voyage is also specified, and the level of cargo in the onshore inventories that service the trades is monitored at all times. An overview of the differences between this thesis and Arnulf and Bjørkli [2010] is provided in Table 1.

This introduction will be concluded by a brief presentation of the rest of the thesis. The following chapter will give insight in planning problems within liner shipping and the fleet deployment problem in particular. Chapter 3 provides a brief review of previously published literature relevant to the problem addressed, and is followed by a thorough description of the problem in Chapter 4. Chapter 5 consists of modeling issues, the mathematical formulation and a description of the heuristic technique used. Issues regarding the input, problem instances, tests and results are presented in Chapter 6. Chapter 7 provides the concluding remarks and suggestions on future work.

2 Liner shipping

Shipping operations today are grouped into three categories. The classification that has become the standard was presented in Lawrence [1972] and separates shipping operations into industrial-, tramp- and liner shipping.

2.1 Industrial-, tramp- and liner shipping

In industrial shipping and tramp shipping, the fleet is deployed based on the current demand at all times, and thus operates dynamic schedules. Industrial shipping differs from tramp shipping by that the industrial shipping companies own both the cargo and the fleet, and thus control the entire chain of operations. Tramp shipping companies operate partially on contracts and partially service the spot market, being available at short notice to any cargo at any port. Typical markets for these areas of shipping are the markets of liquid bulk cargo as oil and liquid natural gas (LNG), and the dry bulk markets of e.g. coal, grain and minerals. Industrial shipping in particular, mainly exist in oil- and gas companies.

Liner shipping companies operate on published schedules, consisting of predetermined routes of ports for loading and unloading. In general, the schedules need to be completed regardless of ship utilization. Consequently long term contracts, planning and demand analysis become important. Liner shipping companies mostly service containerized freight, typically finished consumer products or manufactured parts. The value per volume unit and weight unit of these products are in general higher than for bulk products and reflects the higher cost containerized transport faces compared to the transport of bulk cargo. Lawrence [1972] provides a more in-depth explanation of the characteristics and distinctiveness for the different shipping types.

2.2 Planning in liner shipping

The standard for classifying planning problems within liner shipping separates the problems into strategic-, tactical- and operational problems based on the length of the planning horizon. The strategic planning level incorporates in example route design and fleet size and mix. Which routes to operate, ports to visit, and number and size of ships in the fleet, are long term desicions that take months to change the effects of. In contrast to the strategic problems we find operational planning problems as determining the amount of spot cargo to accept in a port. Cargo-booking and similar decisions have effect on the operations the following couple of days and are short term decisions that are taken on a day-to-day basis.

The fleet deployment problem sorts under tactical planning problems, with a planning horizon spanning from a couple of weeks to a couple of months. Tactical planning problems are based on the decisions from the strategic planning phase. Thus, it is also important for the planning staff with responsibility for the strategic decisions to understand the tactical planning phase to be able to generate the best possible integrated

solution. Fleet deployment is concerned with which ships to allocate to which routes and the sequence of voyages to operate for each ship. The objectives are to maximize the utilization of the fleet and minimize the total costs. The main concerns are which ship types to use on which trade routes due to the properties regarding load capacity and travel speed, and to specify when each ship is scheduled to service the given trade route. A comprehensive review on planning problems within shipping is provided in Christiansen et al. [2007].

In Saga, the planning process is divided into two stages. Saga's headquarter is located in Tønsberg, Norway. This is where the first stage of the process is performed. At the headquarter, costs are estimated and the ships are assigned to voyages based on these estimations. The plan is then sent out to the regional offices, where the operational planning stage may result in modifications due to unforeseen events. This thesis is providing a tool to the headquarter in their planning process. The headquarter basically performs two tasks. First an estimation of time and cost has to be made in order to provide input data for the fleet deployment problem. This estimation is based on information about the fleet and distances between all ports. The information is used to create what may be called template voyages. It is assumed that a vessel services a particular voyage at the same cost and with the same duration every time. Furthermore, the template voyages are sent together with fleet specifications, inventory data and spot cargo availability as input to the fleet deployment problem. The input files used in this thesis are generated the same way based on real data from Saga. When the job of solving the fleet deployment problem is done, the results are sent out to Vancouver, Antwerp or one of the other regional offices. As the regional offices is able to obtain and react to sudden changes that affects the ships, operational changes may be decided. Examples of such sudden events could be a production failure at one port making the ship skip that particular port, a crane breakdown in one port that makes it favorable to visit that port later in the sequence, or a large supply of particularly profitable spot cargo at one given port.

3 Literature review

This section presents an overview of the existing literature on operations research applied to planning problems in the maritime shipping industry.

3.1 Operations research historically

While the airline industry and land based transportation services long have used operation research extensively, operations research has received relatively little attention within the shipping industry. This is reflected in the number of publications in the literature on the respective areas. Ronen [1983], Ronen [1993] and Christiansen et al. [2004] have conducted thorough reviews on the literature published on ship routing and scheduling in the periods 1950-1983, 1983-1993 and 1993-2004 respectively. The first review, Ronen [1983], contains 40 references and reaches back to how Dantzig and Fulkerson [1954] presented how the simplex algorithm may help solve the fleet size problem formulated as a linear programming problem. In contrast does Bodin et al. [1983], published the same year on routing and scheduling of vehicles and crews, contain 700 references. Furthermore, Ronen [1993], which covered the development over the next decade also contained 40 references, while Laporte and Osman [1995] on classical routing problems included exactly 500 references, mainly to publications from the 1980s and early 1990s. Despite the fact that Christiansen et al. [2004] approximately doubled the number of references presenting literature from between 1993 and 2004, it is safe to say that relatively few contributions exist for ship routing and scheduling.

Despite its position as the major artery of international trade, maritime shipping have suffered under the lack of attention from researchers. Even though being presented three decades ago, the explanations presented in Ronen [1983] are still highly relevant. The fact that the dominant transportation modes in the USA are truck and rail is still valid. Thus, shipping receives relatively little attention from the major research communities for quantitative methods. The conservative shipping industry is also an inhibiting factor on new ideas such as the use of optimization models. Still relevant is also the nature of the maritime planning problems. The variety in operating environments and structure of the problems, gives general research less impact compared to the impact on standard vehicle routing and scheduling problems on land and in air. At sea there is also much more uncertainty due to rapidly shifting weather conditions, mechanical problems and strikes that may cause significant delays. To build in slack in the schedules is costly, and thus often limited. This results in that few schedules are in fact completed according to plan. Levy et al. [1977] state that the probability of meeting a planned schedule is as low as 30% [Ronen, 1983].

3.2 The fleet deployment problem

For the fleet deployment problem in particular, Bradley et al. [1977] identify Everett et al. [1972] as the first paper to propose a linear programming model as a solution

approach [Gelareh and Pisinger, 2010]. Boffey et al. [1979] then presented methods for ship scheduling on a North Atlantic route using interactive computer programs and heuristics and Benford [1981] presents a simple procedure for selecting the optimal fleet mix and sea speeds in order to be able to perform the given service level at maximum profitability.

The work on the fleet deployment problem in the following decade is strongly influenced by the Greek researchers A. N. Perakis and N. Papadakis. Perakis identified an artificial constraint in Benford [1981] that led to an increase in costs of 15% and presented an improved formulation and solution method in Perakis [1985]. The main contribution from Perakis and Papadakis [1987a] is the development of detailed and realistic operating cost functions and the sensitivity analyses showing the effects of changes in the cost components. Perakis and Papadakis [1987b] presented a computer program implementing a solution method to the fleet deployment problem modeling some of the costs as random variables with known probability density functions. In the next paper in the sequence, Papadakis and Perakis [1989], continue the development in solving the same non-linear fleet deployment problem and includes multiple destinations.

Perakis and Jaramillo [1991] and Jaramillo and Perakis [1991] still use a detailed and realistic model for the estimation of the operating costs of the liner ships servicing various routes, but is now able to present a linear programming formulation for the liner fleet deployment problem. The approach includes some manipulation of the results to achieve integer solutions and therefore do not guarantee optimal solutions, but Jaramillo and Perakis [1991] suggest two promising mixed linear-integer programming formulations. Perakis and Jaramillo [1991] also make a valuable contribution to the overview of early literature on operations research in liner shipping, presenting a review of the development within interactive computer programs, heuristic optimizing models and mathematical and numerical methods with increasingly more realistic models. An integer-programming model determining the optimal deployment of an existing fleet extending the work in Perakis and Jaramillo [1991] and Jaramillo and Perakis [1991] was presented in Powell and Perakis [1997]. The model was tested on a real liner-shipping problem with constraints regarding predefined routes, service requirements and compatibility constraints and presented substantial savings.

Fagerholt and Lindstad [2000] present a real case from the deployment of supply ships servicing offshore installations in the Norwegian Sea. For an overview of the literature addressing fleet deployment and operation, Perakis [2002] presents the literature present up to 2002. Christiansen et al. [2007] present an overview also including publications from between 2002 and 2007.

Last, and highly relevant to this thesis, Arnulf and Bjørkli [2010] present two mathematical formulations of the fleet deployment problem. While the arc flow formulation does not solve large problems to optimality, the computational results for the path flow formulation present optimal solutions for instances containing six trades, twelve ships and about 70 voyages with a planning horizon of 200 days. Instances with

data from Saga are solved to optimality for planning horizons less than five months. Two path flow models are formulated, both based on a decomposition approach and a priori column generation. Finally, four different path reduction heuristics are introduced to reduce the number of paths generated.

3.2.1 Combined inventory management and routing

Combined inventory management and routing is vital for effective supply chain management. To achieve a successful integration, one of the vital conditions that need to be fulfilled is willingness to share information and data between the producer and the transport company. Thus, the majority of the maritime combined inventory management and routing are found in industrial shipping, where the cargo owner is the only actor and controls both the inventory management and the ships [Andersson et al., 2010]. The liner shipping industry may however also have large benefits from a integrated production and transport plan.

Andersson et al. [2010] present an exhaustive review on combined inventory management and routing in maritime transportation. More than 90 papers, mainly publications from scientific journals and book chapters, are reviewed. An in-depth discussion of the combined problem and a classification of the literature is provided. Among other surveys we find Bertazzi et al. [2008] which present a discussion of the various characteristics of inventory routing problems and creates an understanding for the complexities of inventory routing problems by providing a small deterministic example case. Cordeau et al. [2007] provide a general approach to the combined inventory management and routing problem via the standard vehicle routing problem, but include references to papers focusing on maritime applications.

3.2.2 Solution methods

The most commonly used solution method to the fleet deployment problem is based on a column generation approach, where the routes a generated a priori. Generating all the feasible harbor visit sequences may however lead to a problem size too large to handle and is also time consuming, so approaches only generating promising routes are often used. The routes that improve the overall solution make up the columns in the master problem, while the subproblems consists of determining these routes. The optimal solution is determined by iterating between the master problem and the subproblems until no new columns that lead to further improvement are found. A set partitioning approach based on a priori column generation is used in Fagerholt [1999], Fagerholt and Lindstad [2000] and Christiansen and Fagerholt [2002]. Fagerholt [2001] adds soft time windows to obtain better schedules and also Fagerholt [2004] uses the same two-phase approach, but with an integer programming model in the second phase. The complexity and size of the problem is a function of the number of feasible possibilities each ship has. A common approach has been to use time windows to define the possible service time for a given service, and thus the size of these affects the number of feasible

routes. A large window implies more flexibility and possibilities, while a tight window reduces the number of feasible solutions. The window size affects the complexity of the solution algorithm accordingly [Desrochers et al., 1992]. Window size reduction is the first way discussed in Christiansen and Nygreen [1998a] to eliminate arcs in a network representing feasible routes. An arc is eliminated if it is impossible to service two trade routes consequently within the respective time windows of both, if choosing to travel between them. One of the major advantages of set partitioning models, and a reason for their large share in the literature (40%), is that complex nonlinear constraints easy can be incorporated when generating the columns [Christiansen et al., 2004]. This enables a possibility to use heuristics for the column generation and use of standard optimization software for the set partitioning model.

The two major approaches for modeling the fleet deployment problem are through arc flow models or path flow models [Christiansen et al., 2007]. In an arc flow model a binary variable represents whether a specific ship travels directly from a certain port to another given port. By assigning values to these variables, the routes for each ship are constructed. Additionally the model consists of variables keeping track of time for servicing the voyages and the load on each ship. Thus the feasible schedules are explicitly given through constraints on the arcs. Arc flow models have been used in Fagerholt et al. [2009], and by Christiansen and Nygreen [1998a] and Christiansen [1999] that considered a combined inventory management and ship scheduling problem using a Dantzig-Wolfe decomposition.

A path flow model is based on predefined routes and uses a binary variable to represent whether a specified ship performs a given route. Only feasible routes are considered and a route is equivalent with a full schedule specifying arrival times and load along the route. Path flow models are widely used in papers covering maritime planning problems, starting with Appelgren [1969] that published a path flow formulation which was improved in Appelgren [1971] two years later. The approach used Dantzig-Wolfe decomposition to solve the linear programming problem. Appelgren did however not consider inventory management. This was included in Christiansen and Nygreen [1998b] which used paths giving information about the geographical route, load quantity and start time from each harbor to solve a combined inventory management and ship routing problem with time constraints. A branch-and-bound approach was used to make the solution integer optimal.

When comparing the approaches, the a priori generation of feasible routes may be considered an advantage for the path flow approach as all the constraints regarding feasible schedules are handled outside the mathematical formulation. An advantage with the arc flow model is that the mathematical formulation needs fewer variables [Andersson et al., 2010].

Christiansen et al. [2007] describe models for the network design process for liner shipping and Gelareh and Pisinger [2010] provide a mixed integer programming formulation for a model integrating the network design and fleet deployment process for a liner

service. Fagerholt and Lindstad [2007] present a DSS, TurboRouter, that is currently in use by shipping companies and combines advanced optimization algorithms with possibilities for manual planning. Fagerholt [2004] provides a solution method using pre-generated routes as input to an integer programming formulation of a real liner shipping problem. In Fagerholt [1999] the same real world problem is solved with a three-phase approach starting by generating routes, then combining them and at last using these as columns in a set partitioning formulation solving the integrated fleet size and deployment problem.

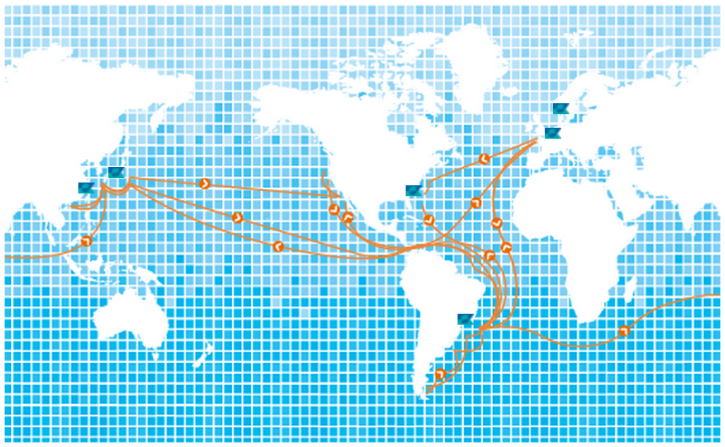

Figure 2: Office locations and trades for Saga Forest Carriers [www.sagafc.com].

4 Problem description

This section will provide a quick recap of important aspects of the fleet deployment problem, before the problem is more explicitly defined by going through the restrictions and properties that require special consideration.

The fleet deployment problem represents the problem of determining how the fleet is to be deployed to a given set of predefined trades. This is equivalent to the problem of determining the sequence of voyages to operate and the schedule for each ship in the fleet. The objective is to define the optimal plan that assigns the ships such that all voyages are served while minimizing cost. Fleet deployment is one of the central tactical planning problems in liner shipping. The trade routes are determined by the liner shipping company after a thorough analysis of the demand in the different regions and then published so that the charterers may enter into contracts regarding transport of goods along the routes. Figure 2 shows the locations of Saga's headquarter and regional offices, and how the trades are spread between the world's major trade markets. The world market of today consists of regions with different properties in terms of labor costs, available technology and access to raw material. This implies that the demand for transport of goods is different between the different regions. Consequently the shipping company often has to sail ballast in order to reposition their ship for its new voyage. It is central for the profitability to minimize the amount of non-profitable ballast sailing.

Charter in, charter out and spot cargo. In this thesis, the fleet deployment problem is mainly considered as assigning the current fleet to voyages. The possibility to charter in ships for certain voyages will be limited to a minimum and is only to be looked upon as an emergency solution. To acquire additional cargo to the ships sailing ballast is however a highly desirable way of generating extra profit without contract obligations,

and is therefore an important aspect of the fleet deployment problem. Additional cargo is only accepted on ballast journeys if time allows it and it is profitable. Excess capacity on regular voyages may also be exploited to generate extra profit by accepting spot cargo in ports of call on the scheduled voyages. This makes the demand for transport of spot cargo in the different ports a relevant issue in the fleet deployment problem, since it may affect the decision in terms of where the larger ships are best utilized.

Fairly evenly spread. In order to achieve the best utilization of the fleet possible, it is favorable to enter into contracts that are flexible regarding visiting times. Flexibility when it comes to restrictions on time of visit enables more alternatives in terms of available ships and makes the shipping company less vulnerable to unpredictable weather conditions at sea. The term fairly evenly spread is therefore commonly used in CoAs. It simply means that the voyages are to be spread in time over the planning period, which is one of the charterer's major concerns as limits on inventory capacity and production rate impose limits on the volume of goods in inventory and when goods are ready for transport. As the other major concern is total volume transported, the shipping company has no restrictions regarding the total number of pickups over a period. In example may a larger ship be able to operate a trade with fewer pickups than a ship with less capacity. This may lower the overall costs for a shipping company with a heterogeneous fleet and will be evaluated in the process of assigning the fleet to voyages.

Demand. The CoA specifies the total volume of goods that is to be transported during a planning period.

Choice of operational speed. The cost and time related to a voyage is dependent on the operational speed on the voyage. The company may reduce costs significantly by reducing speed when time allows it or may be just able to perform an additional voyage by increasing speed on one voyage. UNCTAD [2011] reports a significant reduction in idle tonnage in the container market early in 2011, but the liner companies still deploy their ships at reduced operating speeds. In example the majority of the container lines in 2010 and 2011 ran their Asia-Europe services at only 17 to 19 knots, compared to the normal speeds of 21 to 25 knots. The aim is to reduce fuel expenditure and ship overcapacity, and depending on fuel prices, this is estimated to save the liner companies up to $100 per delivered TEU on major East–West routes.

Inventory. Goods transported on a voyage are loaded from an onshore inventory. This inventory has limitations regarding minimum and maximum levels and a given production rate. Consequently the shipping companies have to visit the inventories in a way that prevents the inventories from exceeding their limits.

In some cases multiple trades may operate out of the same inventory. Dependent on the features of this larger inventory this may offer some extra flexibility to the shipping companies if they are able to exploit the extra capacity.

Overlapping voyages. A special case occurs when a ship may be assigned to a voyage that starts in the same region its previous voyage terminates. In this case it is possible to increase efficiency by starting to load the goods for the forthcoming voyage before having unloaded the current goods completely.

5 Solution methods

This section will continue the previous section by presenting how the issues that need special attention are modeled. Secondly, a step by step explanation of the mathematical formulation is provided. The full mathematical formulation is additionally presented as a continuous version with brief explanations of the restrictions gathered at the end in Appendix A. Ultimately, this section contains a presentation of the ideas behind the heuristics that have been used to improve computational time on the larger test instances.

5.1 Modeling

Modeling the fleet deployment problem there are many features that need special consideration. This section will explain how these are handled when modeling the problem, feature by feature as presented in the problem description.

Charter in, charter out and spot cargo. The charter out possibility offered in this thesis is called voyage charter. The charter agreement specifies the two ports the voyage shall occur between and the ship owner covers the expenses related to operation of the ship. The possibility of chartering out ships on a one-voyage basis is included in the model in the input data by creating a new trade. The demand on this trade is zero, but voyages on this trade may be performed if there is sufficient time in the planning period and doing so has a positive net profit.

One possible scenario is that a competing liner shipping company needs to charter in a ship to perform one of its voyages. If chartering out a ship to meet this request is profitable, it is done. Even more, the possibility of generating some profit from the ballast sailing is desirable. In the case where it exists spot cargo in demand for transport between the two regions of the ballast journey, this is favorable compared to sailing ballast even though the revenue per item may be relatively small as long as the ship is still capable of reaching the start of its next contractual voyage. Modeling this, it is necessary to be aware of the possibility for spot ships to perform voyages on these non-contractual trades and prevent this from happening. In this model this is solved by assigning an infinitely high cost for spot ships to perform these voyages.

The possibility for accepting extra cargo on contractual voyages and the option of executing one of the contractual voyages by a hired-in spot ship is modeled by creating new variables assigned to these opportunities.

Fairly evenly spread. The contracted restriction that states that voyages on the same trade are to be serviced with a specified spread in time can be modeled and solved in different ways. One intuitive method, which is commonly used to define pickup intervals is the use of time windows. A lower and an upper bound define a time window when a voyage can take place in time. Tight time windows ensures a good spread and reduces the solution space and solution time, but does at the same time corrupt

the planning flexibility. A different method, which also can be used in combination with time windows, is to introduce a threshold to state the least allowed difference in time between one voyage and the next on a trade. This prevents voyages that might have been allowed by loose time windows to occur too close in time. The drawback from incorporating a hard restriction like this is that in real life it may be possible to negotiate violations of the minimum spread and thus, the hard restriction might result in lost business opportunities. Furthermore, this may be prevented by allowing the threshold to be violated at a penalty cost. This might result in better solution quality if it for instance results in avoiding expensive ballast sailing against accepting a penalty cost for violating the minimum spread constraint. The challenge when modeling the minimum spread constraint as a soft constraint is to make the penalty cost reflect the actual costs. This thesis uses no time windows, but only a hard restriction on minimum spread between voyages. This choice is made to not corrupt the flexibility in service times for voyages provided in the CoA by introducing artificial constraints.

Demand. The demand restriction is simple, but worth mentioning because of its way of replacing the common restriction in fleet deployment modeling regarding number of voyages on a trade. Commonly a hard restriction defining the exact number of voyages on a trade is calculated from the demand. Using the demand restriction as it is described in the CoA allows more flexibility in terms number of voyages serviced on a trade, by determining the optimal number in the process of solving the fleet deployment problem.

Choice of operational speed. One major operating cost for a liner is fuel. The relationship between the ship speed and the fuel consumption is nonlinear. In order to enable linear programming, but still be able to offer a choice of operational speed, the determination of the optimal speed is separated from the fleet deployment problem and taken into account when generating the routes based on distances and operating costs. This enables the model to offer a choice of a fixed operational speed from a discrete set. The choice has effect on the voyage about to be performed and the following ballast journey. As the point of offering a choice of speed is to offer the opportunity to save cost if available time, to make it possible to reach the deadline of a journey unreachable otherwise or to increase utilization of particular ships, very little extra value is added by offering separate speed choices for voyages and ballast journeys.

Inventory. The far most common practice for companies with issues regarding combined inventory management and ship routing is to separate the problems into two phases [Andersson et al., 2010]. First, the inventory management system produces a cargo plan that includes information about production rate, quantity available at every given time, pickup ports, and capacity limits for the inventory. The routing, or what may be referred to as the fleet deployment relating it to this thesis, is then planned separately in phase two based on the cargo plan. This is also how this thesis models the combined problem. The inventory control is handled by integrating the data for the inventories in the model, such that constraints on the inventories are kept when

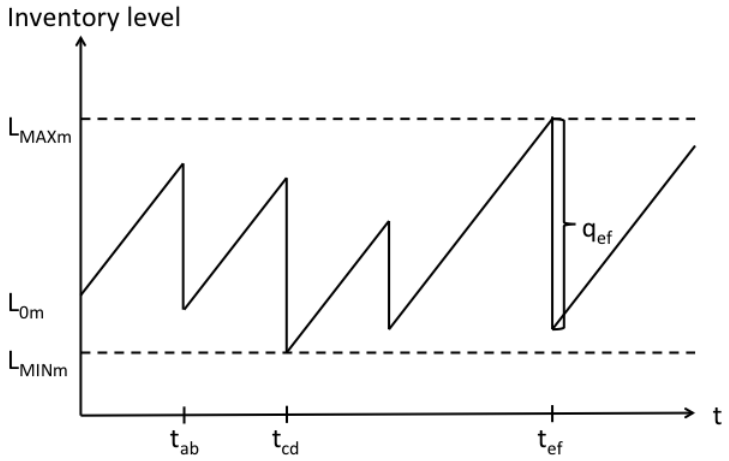

Figure 3: Inventory level for inventory m as a function of time.

developing the solution to the fleet deployment problem. The inventory management is thus a central part when creating a feasible plan for assigning and scheduling ships, and inventory control a necessity for the fleet deployment planners. This thesis handles the inventory issues by restrictions that allows inventory control independent of which ships or trades that operate out of the inventory. The inventory is kept track of by a separate variable for each inentory and upper and lower bounds enforce the ships to service the inventory within these limits. Figure 3 shows how the inventory level may change during a planning period. The drops in inventory level reflects the loading when a voyage is serviced, while the production rate sets the slope for the increase in between voyages. In this case, four voyages is serviced from inventory m during the period. As we can see from the figure, t_{ab} is chosen by convenience for the fleet, while t_{ef} is enforced by the inventory level reaching the maximum limit and the quantity loaded at t_{cd} may be bound by the minimum inventory limit. Assigning every trade to an inventory is solved easy by matrices that keeps track on which trades that operate out of which inventories. Facilitating shared inventories forces the introduction of the variables q_{ripj} and y_{ripj} to be able to keep track of the inventory level when operated by multiple trades. This is described in detail in the mathematical formulation in section 5.2.

Overlapping voyages. In the case where a ship end its voyage in the same geographical region as it is about to start its next, it may be possible to improve efficiency by investigating the possibility of starting to load cargo for the upcoming voyage before all cargo from the previous voyage is discharged. If treating all voyages separately and not considering this opportunity, all ships first will visit all the ports where they shall discharge cargo, and then transit to the first port for loading cargo for the next voyage. It is however a plausible scenario that the ship is scheduled to visit some of the same ports for both voyages. In these cases, if the ship is able to discharge and load on the same visit, it may actually be able to avoid sailing ballast between the two voyages. A prerequisite

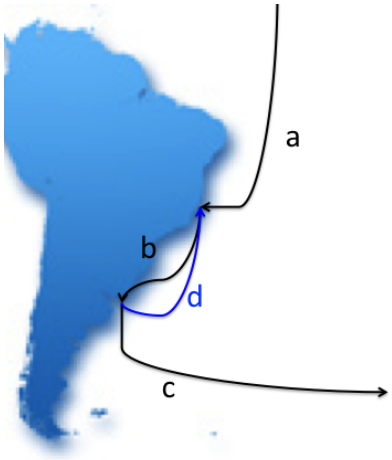

Figure 4: Overlapping voyages may lead to negative ballast sailing costs.

is that the cargo is discharged and loaded such that the ship has the necessary capacity at all times. To optimize the port sequence is however not a part of our model and is done in the cost estimation process carried out before the fleet deployment problem is solved. Hence, is the possibility of overlapping voyages incorporated if it is included in the estimates of cost and duration of ballast sailing between two voyages that may overlap. If two voyages overlap, the ballast sailing cost and time between these may have a negative sign to reflect the cost and time saved. Figure 4 shows why these negative numbers occur. The letters refers to the costs of traversing the arcs representing voyages (black arcs) or the ballast journey (blue arc). The figure shows two subsequent voyages, the first ending in South America, the next starting in the same region. Both voyages is scheduled to visit two ports. If no loading of new cargo can be done until the first voyage is completed, the total costs will be $(a + b) + d + (b + c)$. If the possibility to overlap is included a and c still is performed, but as the ship also loads cargo when it reaches the first port, b only need to be traversed once, and no ballast journey (d) is necessary. This makes the new costs $a + b + c$. As the calculations are done outside the model, the model remains equal, meaning that the overlap has to be included when generating the costs. Hence the ballast journey, d' needs to take the value $-b$, making the total costs $(a + b) + d' + (b + c) = (a + b) + (-b) + (b + c) = a + b + c$ which is the desired outcome.

5.2 Mathematical formulation

This section presents an arc flow formulation that covers all described assumptions and possibilities. An arc flow model can be represented by nodes and arcs. The nodes represent voyages that are linked together by arcs that describes how the ship travel from one voyage to the next. Figure 5 shows a small example for the possible arcs for one ship in a graph consisting of three ships, two trades with two voyages each and no

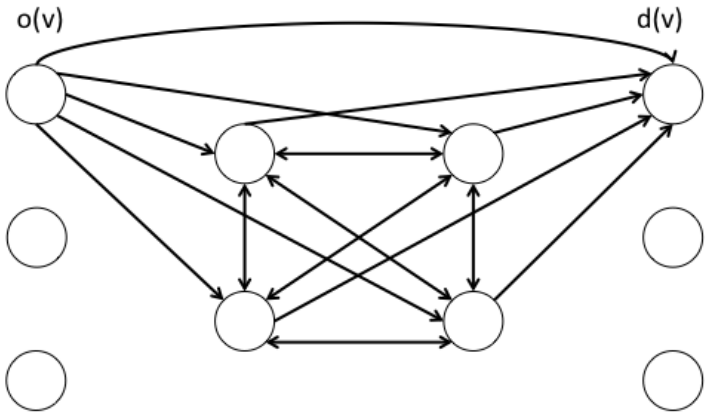

Figure 5: Network example for an arc flow model.

speed option. Note that all the arcs in the graph need to be duplicated for each choice of speed and that each voyage contains different sets of information regarding cost and time, correspondingly. The problem is formulated as a deterministic cost minimization problem. Initially, all necessary indices, sets, parameters and variables are declared. In the literature, *vessel* and *ship* is used interchangeably. In this thesis, ship is used consistently, but to avoid confusion with variables and constants related to spot ships, the letter v will be used to index ships.

Sets:

V	Set of available ships.
R	Set of trades.
N_r	Set of voyages for trade $r, N_r = \{1, 2, ..., n_r\}$.
R_v	Set of trades that can be serviced by ship v.
V_r	Set of ships that can service trade r.
A_v	Set of arcs that can be sailed by ship v.
M	Set of inventories.
R_r	Set of trades that operate from the same inventory as trade r.
K_v	Set of possible speeds for ship v.

Indices:

v	ship, $v \in V$.
r, p	trade, $r, p \in R$.
i, j	voyage, $i, j \in N_r$.
$o(v)$	initial position for ship v.
$d(v)$	final position for ship v.
m	inventory, $m \in M$.
$m(r)$	inventory for trade r.
k	speed, $k \in K_v$.

Parameters:

C_{vrk}	Cost of performing a voyage on trade r by ship v at speed k.
C_r^S	Cost of performing a voyage on trade r by a spot ship.
T_{vrk}	Duration of a voyage on trade r when performed by ship v at speed k.
T_r^S	Duration of a voyage on trade r when performed by a spot ship.
$C^B_{o(v)rk}$	Ballast sailing cost from origin to first voyage on trade r at speed k.
C^B_{vrpk}	Ballast sailing cost from a voyage on trade r to a voyage on trade p at speed k.
$T^B_{o(v)rk}$	Ballast sailing duration from origin to first voyage on trade r at speed k.
T^B_{vrpk}	Ballast sailing duration from a voyage on trade r to a voyage on trade p at speed k.
D_r	Demand on trade r.
Q_v	Maximum load capacity on ship v.
Q^S	Maximum load capacity on spot ships.
G_m	Production rate at inventory m.
W_r	Minimum spread between voyages on trade r.
T	Time at the end of the planning period.
L_{MAXm}	Maximum inventory capacity at inventory m.
L_{MINm}	Minimum limit at inventory m.
L^S_{MAXr}	Maximum available spot cargo on trade r.
L_{0m}	Initial quantity at inventory m.
E_v	Earliest possible start time for ship v.
A_{ri}	Time window opens for voyage i on trade r.
B_{ri}	Time window closes for voyage i on trade r.
P_r^S	Unit revenue for additional spot cargo.

Variables:

x_{vripjk}	equals 1 if voyage i on trade r is serviced right before voyage j on trade q by ship v at speed k, and 0 otherwise.
$x_{o(v)rik}$	equals 1 if voyage i on trade r is serviced first by ship v at speed k, and 0 otherwise.
$x_{rid(v)k}$	equals 1 if voyage i on trade r is serviced last by ship v at speed k, and 0 otherwise.
$x_{o(v)d(v)}$	equals 1 if ship v does not service any voyage, and 0 otherwise.
$t_{o(v)}$	Start time from origin.
t_{ri}	start time for voyage i on trade r.
q_{ri}	Quantity loaded on voyage i.
q_{ripj}	Quantity loaded on voyage (p, j) if performed before voyage (r, i) in time.
q^S_{ri}	Quantity additional spot cargo loaded on voyage i.
l_{ri}	Available inventory on trade r at the start of voyage i.
s_{ri}	equals 1 if voyage i on trade r is serviced by a spot ship.
y_{ripj}	equals 1 if voyage (r, i) is serviced after voyage (p, j) in time.

Objective function:
The problem is formulated as a minimum cost problem, adding all costs related to performing the voyages either by the company's own fleet or by spot ships, and subtracting any profits from accepting extra spot cargo on any voyage.

$$min\,(\sum_{v\in V}\sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} C^B_{o(v)rk}x_{o(v)rik} + \sum_{v\in V}\sum_{r\in R_v}\sum_{i\in N_r}\sum_{p\in R_v}\sum_{j\in N_r}\sum_{k\in K_v}\left(C^B_{vrpk}+C_{vrk}\right)x_{vripjk} + \sum_{v\in V}\sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} C_{vrk}x_{rid(v)k} + \sum_{r\in R}\sum_{i\in N_r} C^S_r s_{ri} - \sum_{r\in R}\sum_{i\in N_r} P^S_{ri}q^S_{ri}\ )$$

Flow constraints:
The following constraints ensures that every ship has a valid path through the node network.

Every ship is required to leave its start position:

$$x_{o(v)d(v)} + \sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} x_{o(v)rik} = 1, v \in V \tag{5.1}$$

Every ship is required to reach its destination node:

$$x_{o(v)d(v)} + \sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} x_{rid(v)k} = 1, v \in V \tag{5.2}$$

Node balance for every voyage node:

$$\sum_{k\in K_v} x_{o(v)rik} + \sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjrik} = \sum_{k\in K_v} x_{rid(v)k} + \sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vripjk}, v \in V, r \in R_v, i \in N_r \tag{5.3}$$

Quantity constraints:
A ship can not accept more cargo than its capacity:

$$q_{ri} + q^S_{ri} \leq \sum_{v\in V} Q_v \left(\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vripjk} + \sum_{k\in K_v} x_{rid(v)k} \right) + Q^S s_{ri}, r \in R, i \in N_r \tag{5.4}$$

A ship can not accept more cargo than currently available in the inventory:

$$q_{ri} \leq l_{ri}, r \in R, i \in N_r \tag{5.5}$$

The contractual demand needs to be satisfied:

$$\sum_{i\in N_r} q_{ri} = D_r, r \in R \tag{5.6}$$

Voyage completion constraints:
One voyage can be serviced one time maximum:

$$\sum_{v\in V}\sum_{k\in K_v} x_{o(v)rik} + \sum_{v\in V}\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjrik} + s_{ri} \leq 1, r \in R, i \in N_r \tag{5.7}$$

To ensure that if one or multiple voyages are skipped on a trade, it is the last one(s):

$$\sum_{v\in V}\sum_{k\in K_v} x_{o(v)r(i+1)k} + \sum_{v\in V}\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjr(i+1)k} + s_{r(i+1)} \leq$$
$$\sum_{v\in V}\sum_{k\in K_v} x_{o(v)rik} + \sum_{v\in V}\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjrik} + s_{ri}, r \in R, i \in \{1, 2, ..., (n_r - 1)\} \tag{5.8}$$

Harbor inventory constraints:
The cargo available at the time of loading for a voyage is given by the following relationship:

$$l_{ri} = L_{0m(r)} + G_{m(r)}t_{ri} - \sum_{j=1}^{i-1} q_{rj} - \sum_{p\in R_r}\sum_{j=1}^{n_p} q_{ripj}, r \in R, i \in N_r \tag{5.9}$$

The cargo available must be within the inventory's limits before and after each voyage:

$$l_{ri} \leq L_{MAXm(r)}, r \in R, i \in N_r \tag{5.10}$$

$$l_{ri} - q_{ri} \geq L_{MINm(r)}, r \in R, i \in N_r \tag{5.11}$$

The following harbor inventory constraints are formulated to set the correct value for q_{ripj} in order to make the calculations for cargo available correct.

The value of q_{ripj} is 0 if (r, i) is equal to q_{ri} if (r, i) is performed after (p, j) and zero otherwise:

$$q_{ripj} \leq q_{pj}, (r, i), (p, j) \in A_v \tag{5.12}$$

$$q_{ripj} \leq \max_{v\in V}\{Q_v\} y_{ripj}, (r, i), (p, j) \in A_v \tag{5.13}$$

$$q_{ripj} \geq q_{pj} - \max_{v\in V}\{Q_v\}\left(1 - y_{ripj}\right), (r, i), (p, j) \in A_v \tag{5.14}$$

The correct values for y_{ripj} are ensured by the following two equations:

$$y_{ripj}\left(t_{ri} - t_{pj}\right) \geq 0, (r, i), (p, j) \in A_v \tag{5.15}$$

$$\left(1 - y_{ripj}\right)\left(t_{pj} - t_{ri}\right) \geq 0, (r, i), (p, j) \in A_v \tag{5.16}$$

Linearized by using the big-M method equation 5.15 can be rewritten as:

$$t_{ri} - t_{pj} \geq M\left(y_{ripj} - 1\right), \text{where } M = T$$

Linearized by using the big-M method equation 5.16 can be rewritten as:

$$t_{pj} - t_{ri} \geq -My_{ripj}, \text{where } M = T$$

Time constraints:
Voyages are required to have a minimum spread in time. Also, the voyages need to be serviced in order, that is voyage 1 before 2 and so on, for every trade:

$$t_{ri} \geq t_{r(i-1)} + W_r, r \in R, i \in \{2, 3, ..., n_r\} \tag{5.17}$$

A ship cannot start its first voyage before reaching it from its starting position:

$$\sum_{k \in K_v} x_{o(v)rik}(t_{o(v)} + T^B_{o(v)rk}) \leq \sum_{k \in K_v} x_{o(v)rik} t_{ri}, v \in V, r \in R_v, i \in N_r \tag{5.18}$$

A ship cannot start its next journey before the previous is completed:

$$\sum_{k \in K_v} x_{vripjk} \left(t_{ri} + T_{vrk} + T^B_{vrpk}\right) \leq \sum_{k \in K_v} x_{vripjk} t_{pj}, v \in V, (r, i), (q, j) \in A_v \tag{5.19}$$

Linearized by using the big-M method equation 5.18 can be rewritten as:

$$t_{o(v)} + \sum_{k \in K_v} T^B_{o(v)rk} x_{o(v)rik} - t_{ri} \leq M_{o(v)ri} \left(1 - \sum_{k \in K_v} x_{o(v)rik}\right),$$

where $M_{o(v)ri}$ is given by: $M_{o(v)ri} = E_{o(v)} + \max_{k \in K_v}\{T^B_{o(v)rk}\}, v \in V, r \in R_v, i \in N_r$

Linearized by using the big-M method equation 5.19 can be rewritten as:

$$t_{ri} + \sum_{k \in K_v} \left(T_{vrk} + T^B_{vrpk}\right) x_{vripjk} - t_{pj} \leq M_v \left(1 - \sum_{k \in K_v} x_{vripjk}\right),$$

where M_v is given by: $M_v = T - E_v, v \in V$

A ship cannot leave its start position before the given time for earliest departure:

$$t_{o(v)} \geq E_v, v \in V \tag{5.20}$$

All voyages must have started before the end of the planning period:

$$t_{ri} \leq T, r \in R, i \in N_r \tag{5.21}$$

Spot market constraints:
A ship has an upper limit on quantity of spot cargo:

$$q^S_{ri} \leq L^S_{MAXr}, r \in R, i \in N_r \tag{5.22}$$

Binary constraints:

$$x_{vripjk} \in \{0,1\}, \quad v \in V_r, (r,i),(p,j) \in A_v, k \in K_v \tag{5.23}$$
$$x_{o(v)rik} \in \{0,1\}, \quad v \in V, r \in R_v, i \in N_r, k \in K_v \tag{5.24}$$
$$x_{rid(v)k} \in \{0,1\}, \quad v \in V, r \in R_v, i \in N_r, k \in K_v \tag{5.25}$$
$$x_{o(v)d(v)} \in \{0,1\}, \quad v \in V \tag{5.26}$$
$$y_{ripj} \in \{0,1\}, \quad (r,i),(p,j) \in A_v \tag{5.27}$$

5.3 Heuristics for large instances

The problem considered in this thesis is a planning problem with a finite time horizon and multiple visits to each customer. The problem can be classified as a Mixed Integer Programming (MIP) problem, and has a problem structure that results in what seems to be an exponential increase in solution time as the number of binary variables increases. Heuristics are a widespread solution approach in the literature. This thesis uses a combination of the exact model and the ideas behind the fix-and-relax heuristic.

One approach that has proven itself effective for solving large instances of MIP problems, like the fleet deployment problem, is fix-and-relax heuristics [Dillenberger et al., 1994]. The basic idea behind this approach is to decompose the problem into subproblems, each subproblem being a partially relaxed and smaller MIP, with a number of binary variables that makes it quick to solve to optimality by the standard optimization software. These subproblems are solved successively, and for each iteration, a new subset of binary variables is permanently fixed to their solution values. Hence, the relaxed variables are gradually reduced in number and eventually disappear. The major advantage of the basic fix-and-relax approach is solution speed, while a weakness is that even if a feasible solution exists, there is no guarantee that the heuristic will be able to find it. For more literature on fix-and-relax methods for MIPs, de Araujo et al. [2007] test the performance of a fix-and-relax heuristic on a lot sizing and scheduling problem formulated as a MIP model. Another recent paper is Beraldi et al. [2008], which present good results for the same type of problem and also shows how fix-and-relax heuristics may be combined with a rolling horizon approach Uggen et al. [2011] present an enhanced fix-and-relax heuristic and applies this to a maritime inventory routing problem. The results presented shows that fix-and-relax heuristics are able to reduce computing time considerably compared to a general Mixed Integer Linear Programming solver, with only a slightly worse objective function value.

Rolling-horizon heuristics follow a similar approach as the fix-and-relax heuristics and are also suitable for overcoming computational infeasibility for large MIP problems. The most common use of rolling-horizon heuristics is in dynamic lot sizing and scheduling problems, where demand gradually becomes known during the planning horizon, but rolling-horizon approaches are also very suitable when all parameters are perfectly known [Mohammadi et al., 2010]. Uggen et al. [2011] presents an enhanced fix-and-relax heuristic and applies this to a maritime inventory routing problem. The results presented

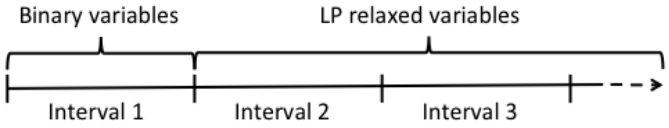

Figure 6: Iteration one in a fix-and-relax heuristic.

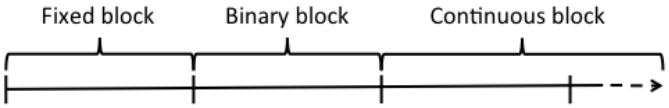

Figure 7: Problem structure for a fix-and-relax heuristic.

shows that fix-and-relax heuristics are able to reduce computing time considerably compared to a general MIP solver, with only a slightly worse objective value.

The fix-and-relax heuristic used as base for the implemented heuristic approaches in this thesis and described in this section is inspired by the basic ideas behind fix-and-relax heuristics given in Dillenberger et al. [1994]. First, the planning horizon is divided into a finite number of intervals, n. A large problem may be decomposed into subproblems in several ways and any index of the integer variables can be used as criterion for partitioning the variables into groups [Ferreira et al., 2009]. The nature of our problem makes the decision variables regarding voyages suitable as separators since the voyages on a trade are performed successively. Furthermore, each interval is considered a subproblem, resulting in n subproblems. In the first iteration, the binary constraints on all decision variables except from the variables concerning a subset of the first voyages for every trade, are relaxed. Relaxing the binary constraints and turning the variables into continuous variables, may be called a partial Linear Programming (LP) relaxation. The next step is to fix all the integer variables from the first iteration to their solution values and reinforce the integrality constraints for the binary variables concerning the next subset of voyages. The problem is then solved iteratively, and the process of fixing variables to their solutions and reinforcing binary constraints are repeated until iteration n, which reinforces the last binary constraints, is completed. The term *block* is commonly used to refer to the intervals and the problem may be grouped into three blocks which we may call *the fixed block*, *the binary block* and *the continuous block*. The first contains the variables that have been fixed to integer values, the second the variables that have been enforced binary constrants, but are not fixed yet, and the last block contains the LP relaxed variables. Inspired by Uggen et al. [2011], but modified and customized to this thesis, Figure 6 and Figure 7 shows the first iteration and the problem structure graphically.

6 Computational studies

The implementation of the mathematical model has been written in the programming language Mosel. The models has the been run with use of the commercial optimization software Xpress IVE on a HP Intel Core i7-2600, CPU 2x3.40 GHz, 16 GB RAM running on Windows 7 SP1.

The full mathematical model presented in section 5.2 is implemented. Regarding constraint 5.15, 5.16, 5.18 and 5.19, the implemented versions are the linearized ones.

The Mosel code is provided in Appendix B.

6.1 Input

All test cases are built on actual data from Saga, where the data obtained has been applicable. The following sections will present the shipping company, the structure of the input data and how data that was not available from Saga was estimated.

Saga Forest Carriers. Saga Forest Carriers Intl AS is an international shipping company specializing in the transportation of forest products and break-bulk cargo (general cargo). Saga is headquartered in Tenvik, situated near Tønsberg, Norway and has regional offices in Vancouver, Savannah, Rio de Janeiro, Antwerp, Tokyo and Shanghai. Saga is a part of the Hesnes Group [www.sagafc.com].

The present fleet consists of 24 sophisticated open-hatch ships. It is a homogenous deep-sea fleet, allowing for flexibility and full interchangeability between ships within the trades that Saga operates. Saga has recently fulfilled an extensive newbuilding program by taking delivery of nine sophisticated open hatch gantry ships from Oshima Shipbuilding Ltd, Japan. Each ship is fitted with two specialized gantry cranes of 40-42 MT lifting capacity. Further, Saga has two ships on order that will be entering the Saga pool within 2014 [www.sagafc.com].

All data reagarding ships is real data provided by Saga. V, the set of available ships varies between the test cases, but all combinations of ships used are subsets of the actual fleet. K_v, the set of possible speeds for ship v, is in every instance a set of two operational speeds, named "high" and "low" and being 20 knots and 14 knots respectively. Maximum cargo capacity on each ship, Q_v is actual data retrieved from Saga. As is E_v, earliest possible start time for each ship, which is set from the situation for Saga's fleet at a given point in time. Table 2 lists Saga's entire fleet divided into classes. All ships within the same class are as close as identical. The small differences in cargo capacity and cost are however specified in the input data. The capacity is measured in deadweight tonnage (DWT) and states how much weight a ship can safely carry measured in tonnes. The abbreviation column gives the two letter abbreviation that is used in the input file. The largest instance used in the testing makes use of the entire fleet.

Table 2: The fleet of Saga Forest Carriers.

Ship	**Capacity (DWT)**
Adventure class	
Saga Pioneer	
Saga Odyssey	
Saga Navigator	
Saga Journey	
Saga Frontier	46 882
Saga Explorer	
Saga Enterprise	
Saga Discovery	
Saga Adventure	
Bird class	
Saga Viking	46 882
Saga Voyager	46 882
Saga Andorinha	47 000
Saga Tucano	47 032
Saga Jandaia	47 027
Saga Beija-Flor	46 990
Tide class	
Saga Sky	47 034
Saga Horizon	47 016
Saga Wind	47 053
Saga Spray	47 076
Saga Crest	47 016
Saga Wave	47 062
Saga Tide	47 029
Mitsui class	
Saga Morus	56 800
Saga Monal	

Trades. The set of trades, R, is obtained from the trades that Saga operates. The demand on a trade, D_r, is based on typical numbers from Saga's contracts. The number of voyages is set in accordance to the demand such that the smallest ship in the fleet has the capacity to fulfill the total demand on a trade if servicing all the voyages. Mathematically, the number of voyages on a trade can be formulated like this:

$$n_r = \lceil \max_{v \in V} \{D_r / Q_v\} \rceil, r \in R \tag{6.1}$$

Port constraints as cargo handling equipment available and limits on the size of ships manageable, although liner ships usually are of moderate sizes, affect the compatibility between ports and ships. The fleet of Saga is heterogeneous, but in terms of ship-port compatability the fleet offers full interchangeability between ships for all the ports included in all trades. Hence, R_v includes all trades for every ship and V_r includes all ships for every trade.

W_r, the minimum spread between the voyages on trade is taken from typical numbers in Saga's CoAs.

Cost and time The largest matrices in the input file are the matrices covering costs and durations of voyages and ballast journeys. All the data in these matrices are real data obtained from Saga. Based on the specified set of trades and ships, real data are obtained both for "high" and "low" speed. Ballast costs and sailing duration from origin to first voyage is derived from the ship's starting position. Ballast costs and sailing duration for the journey after a ship's last voyage are all zero. That is, a ship's final position is set to be the final port of its last voyage. As no maintenance schedule is incorporated, this is a fair assumption. All costs are given in US Dollars (USD), while the duration of a voyage is given in days.

Inventory All inventory data has been calculated from the same set of formulas. For all trades, initial quantity, L_{0m}, is set to the capacity of the Mitsui class, the class with the largest capacity. The minimum and maximum limits at every inventory, L_{MINm} and L_{MAXm}, are set to 0 and two times the capacity of the Mitsui class respectively. The production rate, G_m, is calculated for every instance by using the built-in solver in Microsoft Excel to find the rate such that the quantity at the end of the planning period is equal to the initial quantity. Hence total quantity produced over a planning period equals demand for every inventory. When multiple trades share one inventory, the individual inventory data from the involved trades is summarized into one large inventory. The following equation ensures that an inventory does not exceed its capacity if the quantity given by the demand is transported within time:

$$L_{0m(r)} + G_{m(r)}T \leq L_{MAXm(r)} + D_r + \sum_{p \in R_r} D_p, r \in R \tag{6.2}$$

Spot ships To estimate the voyage cost and voyage time for using a spot ship, a ship from Saga's Adventure class has been used as a template. The load capacity and duration of performing a voyage is set to be equivalent, as we assume that when planning ahead a spot ship is available at the desired time. To reflect the additional cost of hiring a spot ship to perform a voyage a penalty of 200% has been added. The penalty is estimated to reflect the positioning costs, the voyage cost and the profit margin demanded by the spot ship.

Spot cargo Data for the market for spot cargo on each trade is difficult to estimate as the market is volatile and no real data is obtained. In this thesis it is assumed that

Table 3: Test instances.

Instance	No. of ships	No. of trades	No. of voyages 30d	60d	90d	120d	150d
(4S, 3T, xxD)	4	3	4	8	12	16	20
(8S, 3T, xxD)	8	3	6	12	19	25	31
(10S, 4T, xxD)	10	4	7	14	21	28	35
(12S, 5T, xxD)	12	5	9	18	27	36	45
(15S, 6T, xxD)	15	6	11	22	34	45	56
(15S, 8T, xxD)	15	8	11	22	33	44	55
(20S, 8T, xxD)	20	8	14	28	42	56	70
(25S, 10T, xxD)	25	10	17	34	51	68	85

spot cargo is available in every port at any time. The maximum limit for spot cargo on a voyage, L^S_{MAXr}, is set to 20% of the capacity of a ship in the Adventure class to prevent spot cargo from becoming the main purpose of a voyage. The unit revenue is derived from calculating the unit costs for transporting contractual cargo.

Planning period The time at the start of the planning period is zero. The time at the end of the planning period, T, therefore also states the length of the planning period. Typically today, a planning period of three months is used by the planners in the shipping companies. In most of the test instances, the typical length of 90 days are used, but shorter and longer horizons were part of the test instances for testing performance on various sized problems. The horizon of the planning period is given, as voyage durations, in days.

6.2 Instances

The testing is executed on a set consisting of 40 instances in total. The base is a set of eight instances where the number of ships in the fleet and number of trades is varied. These eight instances are then run with five different planning horizons with differences in duration. Table 3 shows the composition of ships and trades, and number of voyages in total for the different instances. The first column also states how the instances are named and referred to in the rest of this thesis. The name of the instance gives the following information: number of ships in the fleet, number of trades, duration of the planning period measured in days. For example, the smallest instance with four ships and three trades will be denoted *(4S, 3T, 90D)* for the typical planning period of 90 days. *No. of Voyages* is given by equation 6.1 and states the number of voyages available to fulfill the commitments in the CoA.

Table 4: Example instance.

Fleet			
Ship	**Abbr.**	**Class**	**Capacity**
Saga Adventure	AD	Adventure	46 882
Saga Andorinha	AN	Bird	47 000
Saga Beija-Flor	BJ	Bird	46 990
Saga Monal	ML	Mitsui	56 800
Trades			
Origin-destination	**Abbr.**	**Demand**	**No. of voyages**
Europe-US East Coast	EUREC	160 000	4
South America-Europe	SAMEUR	170 000	4
South America-Far East	SAMFE	165 000	4

6.2.1 Example instance

To describe an example of a problem instance we use *(4S, 3T, 90D)*. The most important data for the example instance is provided in Table 4. It is also worth mentioning that even though the three trades in this particular problem has the same number of voyages, large variation occur in different instances. In example *(8S, 3T, 90D)* has 13, 3 and 3 voyages for its trades which effects the complexity of the problem. As the table shows, the numbers of voyages are given such that the planners may choose to service the entire trade with *Saga Adventure*, the smallest ship.

$$4 * 46882 = 187528 \geq max\{160000, 170000, 165000\} \tag{6.3}$$

This quick calculation shows that four voyages with the smallest ship is sufficient to meet the demand on every trade. We also see that if *Saga Monal* is assigned to all the voyages on one trade, only three voyages are necessary to satisfy the demand, as three times *Saga Monal's* capacity exceeds the demand of every trade:

$$3 * 56800 = 170400 \geq max\{160000, 170000, 165000\} \tag{6.4}$$

We also see that a composition of ships where *Saga Monal* is used twice to service the *EUREC* trade and *Saga Adventure* once (as *Saga Adventure* is the smallest ship in the fleet, any other may also be used), is sufficient to satisfy the demand for the EUREC trade of 160 000 DWT.

$$2 * 56800 + 46882 = 160482 \geq 160000 \tag{6.5}$$

As SAMEUR and SAMFE both operate from South America in this example they also operate from the same inventory. The origins and destinations for the trades, and the locations of inventories are shown in Figure 8. The optimal solution to the

Figure 8: Trades and inventory locations for the example instance.

fleet deployment problem for this instance is presented in Figure 9. Note that the fourth voyage on the SAMEUR trade is not serviced by any ship, as the three first voyages on the trade are serviced by *Saga Monal* which is sufficient to meet the demand requirement. Choice of speed is however not included in the graphic representation of the solution.

The following two sections will present all the results obtained from the testing. The first section will present the test results related to the performance of the model on differently sized instances, while the second section compares the formulations developed in this thesis to other approaches for modeling constraints in the fleet deployment problem. The conclusions drawn from the testing are presented in Chapter 7.

6.3 Testing performance

The first set of tests is constructed to test the limits in terms of number of ships and trades, and length of planning horizon, for which the full model still returns valuable results for a planner. The tests for the full model is run on all 40 instances generated. Then a variety of heuristic approaches is tested to investigate the value of heuristics as the problem sizes increase. Furthermore, the effects of the problem structure for individual problem instances are analyzed. The unit for the objective function values, USD, is denoted $.

6.3.1 Performance of the full model

The first set of tests examines how the full model performs in terms of solution quality and computational time.

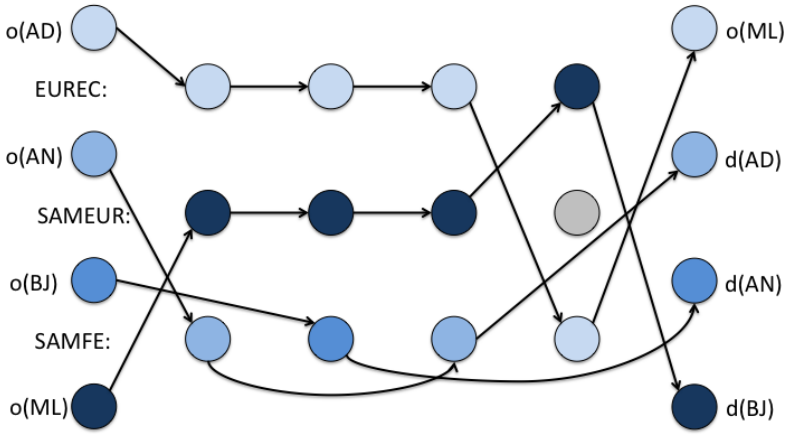

Figure 9: Optimal solution for the example instance.

Table 5: Results for the full model (30 and 60 days planning horizon).

	30 days			60 days		
Instance	**Obj.val.($)**	**Sol.time**	**Gap**	**Obj.val.($)**	**Sol.time**	**Gap**
(4S,3T)	2 455 520	0.02s	0 %	2 914 385	0.41s	0 %
(8S,3T)	1 922 790	0.05s	0 %	4 146 371	63.33s	0 %
(10S,4T)	1 764 481	0.06s	0 %	3 974 976	21.83s	0 %
(12S,5T)	2 304 396	0.11s	0 %	4 749 197	5.69s	0 %
(15S.6T)	2 701 647	0.25s	0 %	4 734 188	3.77s	0 %
(15S,8T)	3 736 622	0.58s	0 %	5 470 434	1h	5.1 %
(20S,8T)	3 949 610	0.41s	0 %	8 680 322	1h	3.2 %
(25S,10T)	5 256 585	1.44s	0 %	9 112 020	1h	9.3 %

As we can see from Table 5, the full model solves all instances to optimality for a planning horizon with a duration of 30 days within two seconds. Optimal solution is also guaranteed for five out of eight instances when the planning period is extended to 60 days. For *(15S,8T,60D)* and the two larger instances, the model starts to struggle with guaranteeing that optimal solution is found. The model was terminated and forced to return the solution after one hour to make the results regarding gap comparable with the rest of the tested instances. In Table 6, only solutions detected within one hour are displayed. If no feasible integer solution was found within the hour, the model was set to terminate. As we see from the table, planning horizons of 120 and 150 days resulted in solutions with large gaps between optimal solution and best bound, and for the largest instances more computational time is needed to detect feasible integer solutions. The typical length of the planning period used in the industry is 90 days. When comparing the full model to all the models with variations in terms of e.g. inventory control, time windows, speed choice or number of voyages, results from running the full model for one

Table 6: Results for the full model (120 and 150 days planning horizon).

	120 days			150 days		
Instance	**Obj.val.($)**	**Sol.time**	**Gap**	**Obj.val.($)**	**Sol.time**	**Gap**
(4S,3T)	7 631 941	15m 39s	0.4 %	11 309 037	1h	20.4 %
(8S,3T)	8 776 486	1h	22.8 %	10 400 570	1h	20.0 %
(10S,4T)	9 297 279	1h	9.4 %	14 131 901	1h	20.3 %
(12S,5T)	14 266 789	1h	28.0 %	23 840 905	1h	40.4 %
(15S.6T)	18 652 738	1h	26.9 %	-	-	-

Table 7: Results for the full model with 90 days planning horizon.

	1h sol.time	**Unlimited solution time**			**Difference**
Instance	**Obj.val.($)**	**Obj.val.($)**	**Sol.time**	**Gap**	**Obj.val.($)**
(4S,3T,90D)	6 541 743	6 541 743	2m 16s	0 %	0.0 %
(8S,3T,90D)	6 597 846	6 597 846	3h 2m	10.9%	0.0 %
(10S,4T,90D)	6 577 028	6 378 627	4h 24m	2.3 %	-3.0 %
(12S,5T,90D)	7 933 607	7 881 225	9h 26m	6.4 %	-0.7 %
(15S,6T,90D)	10 993 465	10 993 465	1h 54m	5.7 %	0.0 %
(15S,8T,90D)	14 268 801	13 806 533	3h 25m	33.6 %	-3.2 %
(20S,8T,90D)	15 866 013	14 005 143	11h 45m	7.0 %	-11.7 %
(25S,10T,90D)	37 252 542	16 512 932	27h 33m	13.7 %	-55.7 %

hour is used to obtain a good basis for comparison. Table 7 shows the results compared to the benchmark values, when the model is allowed more computational time. We see that more computational time provide better solutions, both in terms of objective function value and gap for most instances.

6.3.2 Fix-and-relax heuristics

Six variations of a fix-and-relax heuristic have been implemented and tested. Table 8 shows the specifications for four of the heuristics, while the instance specific specifications for the heuristic 5 are presented with its results in Table 11 and the last variation is described in the text. All heuristics can be classified as two-block fix-and-relax heuristics.

The four rows in Table 8 named by type of flow variable, describes whether a variable is continuous, binary or removed in the first iteration, or what we define as Block 1. The first voyage on a trade is abbreviated *v1*, the second *v2* and further on. That *v1* is set as binary means that x_{vripjk} is a binary variable when $i = 1$. An exeption is found in heuristic 2, where a binary constraint for a voyage means that $j = 1$ in the flow variable, x_{vripjk}. $j = 1$ can be interpreted as the incoming arc to voyage j, and a *-notation is used to specify when this is the case. Note that $x_{o(v)d(v)}$ is removed from the model

Table 8: Specifications for heuristic 1-4.

	Heuristic 1	**Heuristic 2**	**Heuristic 3**	**Heuristic 4**
			Block 1	
x_{vripjk}	v1, v2 binary	v1 binary*	v1 binary	v1, v2 binary
$x_{o(v)rik}$	Binary	Binary	Binary	Binary
$x_{rid(v)k}$	Continuous	Binary	Continuous	Binary
$x_{o(v)d(v)}$	Removed	Removed	Removed	Removed
Time limit voyages	-	-	50 days	50 days
Voyages per ship	2	-	2	-
First voyage priority	-	-	Yes	Yes
			Block 2	
x_{vripjk} fixed	Voyage 1	Voyage 1*	Voyage 1	Voyage 1
$x_{o(v)rik}$ fixed	Yes	Yes	Yes	No

for every heuristic, including heuristic 5 and 6. This was done to prevent a large part of the fleet to go idle when solving the first iteration, while a few ships performed all voyages with binary constraints. Activating the entire fleet is necessary when solving the problem instances in this thesis, as every instance is tight in terms of high demand for transport relative to the horizon of the period. The row named *Voyages per ship* in Table 8 refers to a constraint enforced to ensure that no ship performs more than a given number of the voyages with binary constraints in the first iteration, while *Time limit voyages* ensures that no ship is performing the voyages too late in the period for the rest of the voyages on the trade to be completed. *First voyage priority* is stating that a ship has to service the first voyage on a trade as its first voyage after leaving from its origin. For heuristic 5 and 6, $x_{o(v)rik}$ and $x_{rid(v)k}$ are binary, $x_{o(v)d(v)}$ is removed and the *First voyage priority* is enforced. The rows regarding Block 2 states which variables that are fixed from the solution of the first iteration when solving the second iteration where, since we only use two iterations, all variables that originally were so, are binary.

How heuristic 1-6 performs in comparison to the full model is presented in Table 9-12. The values in the *Diff.* columns refer to the heuristics' differences in objective function values compared to the corresponding values for the full model. The first issue worth noticing is that every instance not larger than *(16S,6V,90D)* is solved in less than one minute. The larger instances need a couple of minutes, but still provides substantial improvements from the full model, which was not able to guarantee an optimal solution to any of the larger instances in an hour. The objective function values are however for most instances, substantially worse when comparing to the full model. The exception are some of the largest instances, where actually significant improvements are seen. These results occur because of the reduced solution time, which allows the heuristic to reach its optimal solution, while the full model returns the best solution found after one hour, which may not be an overall good solution at all.

Table 9: Results for heuristic 1 and 2.

	Heuristic 1			**Heuristic 2**		
Instance	**Obj.val.($)**	**Sol.time**	**Diff.**	**Obj.val.($)**	**Sol.time**	**Diff.**
(4S,3T,90D)	9 753 208	1.5s	49.1 %	8 581 818	2.4s	31.2 %
(8S,3T,90D)	10 817 439	0.9s	64.0 %	9 246 883	4.4s	40.2 %
(10S,4T,90D)	8 370 162	9.8s	27.3 %	7 204 685	6.6s	9.5 %
(12S,5T,90D)	14 138 288	13.1s	78.2 %	8 957 737	51.2s	12.9 %
(15S,6T,90D)	12 464 189	32.9s	13.4 %	14 147 074	14.2s	28.7 %
(15S,8T,90D)	11 043 643	2m 44s	-22.6 %	10 519 756	2m 53s	-26.3 %
(20S,8T,90D)	19 456 754	1m 51s	22.6 %	16 122 629	2m 18s	1.6 %
(25S,10T,90D)	27 231 040	3m 19s	-26.9 %	20 271 887	15m 47s	-45.6 %

Table 10: Results for heuristic 3 and 4.

	Heuristic 3			**Heuristic 4**		
Instance	**Obj.val.($)**	**Sol.time**	**Diff.**	**Obj.val.($)**	**Sol.time**	**Diff.**
(4S,3T,90D)	8 188 178	1.1s	25.2 %	8 255 982	1.2s	26.2 %
(8S,3T,90D)	9 988 996	3.3s	51.4 %	6 963 185	9.2s	5.5 %
(10S,4T,90D)	7 927 982	6.2s	20.5 %	6 943 882	47.5s	5.6 %
(12S,5T,90D)	10 454 514	18.0s	31.8 %	8 217 533	1m 9s	3.6 %
(15S,8T,90D)	10 744 053	57.4s	-24.7 %	10 577 865	3m 13s	-25.9 %

Heuristic 5 is the last version of the fix-and-relax heuristics not distinguishing between trades. It is quite similar to heuristic 1-4, but Table 11 needs some explanation for the three rightmost columns. *B1* and *B2* refers to which voyages that are binary restricted and fixed in iteration one and two respectively. The column named $x_{o(v)rik}$ can take two values. If the values of the variable is fixed in iteration two, the value is *F*, while a cell takes the value *V* if the final value of the variable is to be determined in the second iteration. Heuristic 6 is designed based on the results from heuristic 1 through 5. These heuristics were able to reduce the solution time significantly, but returned relatively bad objective function values. The aim when designing this last heuristic was thus to

Table 11: Results for heuristic 5.

	Heuristic 5					
Instance	**Obj.val.($)**	**Sol.time**	**Diff.**	**B1**	**B2**	$x_{o(v)rik}$
(8S,3T,90D)	7 062 132	12.1s	7.0 %	1-3	1	F
(10S,4T,90D)	6 561 822	50.1s	-0.2 %	1-2	1	V
(12S,5T,90D)	8 511 014	33.5s	7.3 %	1-2	1	V
(15S,8T,90D)	10 729 490	1m 59s	-24.8 %	1	1	V

Table 12: Results for heuristic 6.

	Heuristic 3		
Instance	**Obj.val.($)**	**Sol.time**	**Diff.**
(8S,3T,90D)	6 961 527	7.9s	5.5 %
(10S,4T,90D)	6 616 902	1m 38s	0.6 %
(12S,5T,90D)	8 014 312	4m 30s	1.0 %
(15S,6T,90D)	11 115 520	7m 23s	1.1 %
(15S,8T,90D)	11 492 973	1m 33s	-19.5 %
(20S,8T,90D)	14 574 591	9m 57s	-8.1 %
(25S,10T,90D)	40 715 362	8m 56s	9.3 %

Table 13: The best results obtained from the heuristics.

Instance	**Version**	**Obj.val.($)**	**Sol.time**	**Diff.**
(8S,3T,90D)	Heuristic 6	6 961 527	7.9s	5.5 %
(10S,4T,90D)	Heuristic 5	6 561 822	50.1s	2.9 %
(12S,5T,90D)	Heuristic 6	8 014 312	4m 30s	1.7 %
(15S,6T,90D)	Heuristic 6	11 115 520	7m 23s	1.1 %
(15S,8T,90D)	Heuristic 2	10 519 756	2m 53s	-23.8 %
(20S,8T,90D)	Heuristic 6	14 574 591	9m 57s	4.1 %
(25S,10T,90D)	Heuristic 2	20 271 887	15m 47s	22.8 %

allow some more solution time, but achieve better objective function values. This was done by only applying binary constraints to the first voyage on half the trades in the first iteration. This results in a less constrained problem in the second iteration, with fewer variables already determined, and Table 12 shows that objective function values close to, if not better, the values of the full model were obtained. We will conclude the presentation of results from the testing of different heuristics by providing a table that shows the best results obtained from all the heuristics. When using heuristic approaches there is no guarantee of finding the optimal solution or even a feasible solution at all. An approach using multiple heuristics with different properties may therefore provide an increased probability of reaching a good solution. In our case, we see that if all heuristics are run when solving the fleet deployment problem for an instance, the total solution time still is acceptable. If using the approach of running all heuristics and storing the best value, the results presented in Table 13 are obtained.

Actually improving the objective function value for two instances and detecting solutions that are about 7% worse than the full model for the other two, shows that a carefully designed fix-and-relax heuristic is able to find good solutions in very short time compared to the full model.

Table 14: Voyage distribution between trades.

Instance	**Voyages**	**Voyages per trade**
(8S,3T,90D)	19	13, 3, 3
(10S,4T,90D)	21	3, 10, 3, 5
(12S,5T,90D)	27	3, 10, 3, 4, 7
(15S,6T,90D)	34	3, 10, 7, 3, 4, 7

Table 15: Problem structure effects for the full model and when speed is fixed.

	Full model	**Shared inventory**	**Fixed speed**
Instance	**Gap**	**Gap**	**Gap**
(8S,3T,90D)	13.8 %	13.8 %	10.5 %
(10S,4T,90D)	7.4 %	8.8 %	4.4 %
(12S,5T,90D)	8.0 %	7.1 %	3.2 %
(15S,6T,90D)	6.0 %	8.4 %	7.8 %

6.3.3 Problem structure effects

The results in Table 5-7 clearly show that increasing problem size, in general implies longer solution time. Another connection that appears in the results, but is worth highlighting is how the problem structure effects the result. Structure in this context refers to how the number of voyages is distributed between the voyages on a trade. Table 14 shows that instance *(8S,3T,90D)* has one trade with 13 voyages and two trades with three voyages each. This makes *(8S,3T,90D)* the instance with the most extreme structure regarding difference in number of voyages for the trades. Table 15 and Table 16 show that even though *(8S,3T,90D)* has less voyages in total when we compare it to the three larger instances, solution time is higher and the gap is larger, given the allocated computational time of one hour.

Table 16: Problem structure effects for time windows and fixed number of voyages.

	Time windows		**Fixed number of voyages**	
Instance	**Sol.time**	**Gap**	**Sol.Time**	**Gap**
(8S,3T,90D)	81.0s	0 %	1h	16.1 %
(10S,4T,90D)	1.2s	0 %	52m 13s	0.0 %
(12S,5T,90D)	3.5s	0 %	1h	1.9 %
(15S,6T,90D)	57m 13s	0 %	1h	4.5 %

6.4 Testing the performance relative to other modeling solutions

This section describes how the solutions implemented in the full model was measured against other methods of modeling constraints in the fleet deployment problem.

6.4.1 Time windows replacing inventory control

A common approach to ensure fairly evenly spread voyages over the planning period is to use time windows. Hard time windows are absolute and define an interval in which a voyage has to take place, while soft time windows define the same interval, but allows voyages to take place outside the time interval against a penalty cost. This thesis use no time windows at all, but rely on inventory control and a hard minimum spread limit to ensure that all voyages are fairly evenly spread on a trade. The aim of this test is to investigate how the contract requirement that states that all voyages on a trade should be fairly evenly spread in time is best handled.

To enforce hard time windows, we need to define two new parameters. W_{MINri} gives the lower time limit for a voyage to start and W_{MAXri} gives the upper time limit, resulting in $[W_{MINri}, W_{MAXri}]$ being the interval we call the time window. The parameters are present in the following two new constraints:

$$t_{ri} \geq W_{MINri}, r \in R, i \in N_r \tag{6.6}$$

$$t_{ri} \leq W_{MAXri}, r \in R, i \in N_r \tag{6.7}$$

Constraint (6.6) states that all voyages must start after the time window opens, and constraint (6.7) that all voyages must start before its time window closes. At the same time, all constraints regarding inventory control are removed. This applies to constraint (5.5) regarding quantity loaded and the harbor inventory constraints (5.9), (5.10), (5.11), (5.12), (5.13), (5.14), (5.15) and (5.16). The same constraints are added to, and removed from the code when running the test instances.

The time windows are defined by taking the duration of the planning period measured in days and divide it by the number of voyages for the specific trade. Then the time windows are given a length corresponding to the double of this number and will start at intervals corresponding to this number. For example the first voyage on a trade with three voyages, given the standard planning period of 90 days, will have to start within the interval $[0, 60]$, the second voyage within $[30, 90]$ and the last within $[60, 90]$. Note that the time windows overlaps and that the last window will have half the length of the others as the end time can not be exceeded. This way of defining the intervals makes the time windows quite wide. Tighter time windows were also tested, but with difficulties in obtaining feasible solution. The results from the testing of time windows show that it is hard to obtain good results with time windows. If the charterer defines absolute time windows in the CoA, the planner has no choice, but if the charterer is more flexible it is hard to exploit this flexibility by using time windows. As Table 17, the objective function values are substantially worse when time windows are enforced. Only for the

Table 17: Objective function values when using time windows.

	Full model	**Time windows**		**Spot ships**	
Instance	**Obj.val.($)**	**Obj.val.($)**	**Difference**	**Obj.val.($)**	**Difference**
(4S,3T,90D)	6 541 743	6 169 012	-5.7 %	25 855 363	295 %
(8S,3T,90D)	6 597 846	11 453 077	73.6 %	26 333 536	299 %
(10S,4T,90D)	6 577 028	16 735 943	154.5 %	38 756 272	489 %
(12S,5T,90D)	7 933 607	30 353 481	282.6 %	53 385 125	573 %
(15S,6T,90D)	10 993 465	31 398 801	185.6 %	63 824 004	481 %
(15S,8T,90D)	14 268 801	33 286 355	133.3 %	70 436 550	394 %
(20S,8T,90D)	15 866 013	43 797 335	176.0 %	81 079 136	411 %
(25S,10T,90D)	37 252 542	60 411 370	62.2 %	97 988 296	163 %

smallest instance does the time windows open for solutions not possible when inventory is controlled. The two rightmost columns show the objective function values for servicing all voyages with spot ships and how these values compare to the values for the full model. Using spot ships to perform all voyages will always be a feasible solution as long as number of voyages times minimum spread between voyages does not exceed the planning horizon, and tells us something about the performance of an model compared to what we may call the worst case solution. With reference to the large instances in Table 6 where no feasible integer solution was found within the given solution time, the trivial solution of only using spot ships is a feasible solution also for these. The reason this solution is not detected within the time given is that, as Table 17 shows, the solution where spot ships are used to service all voyages has an objective function value so bad that it is not yet investigated when the algorithm terminates. The algorithm is designed to detect the best solution and the bounds for this, and thus bad solutions as the spot ship solution are not investigated until very late in the solution process. The rigidity of the time windows is an advantage when it comes to solution times, though. As Table 18 shows, the optimal solution is guaranteed in less than one hour for all the instances up to and including *(15S,6T,90D)*. With the time windows used, this is however of little value when the objective function values are of the quality presented, so to find the right size of the interval specifying the time windows is essential for success if choosing that method of modeling. Soft time windows can be used to allow more solutions, against a penalty cost, but still the intervals will be essential, and enforcing a penalty function will compromise the good solution times.

6.4.2 Fixed number of voyages

This thesis offer flexibility in terms of number of voyages on an trade. In contrast to a fixed number, it is determined by the demand and the capacity of the ships assigned to the voyages. This test investigates how our way of modeling voyages per trade performs compared to fixing the number of voyages to a specified number based on the capacity of the smallest ship in the fleet and the demand on a trade. The number of voyages on

Table 18: Solution time and gap when using time windows.

	Full model		Time windows	
Instance	**Sol.time**	**Gap**	**Sol.time**	**Gap**
(4S,3T,90D)	136.1s	0.0 %	8.5s	0 %
(8S,3T,90D)	1h	13.8 %	1m 21s	0 %
(10S,4T,90D)	1h	7.4 %	1.2s	0 %
(12S,5T,90D)	1h	8.0 %	3.5s	0 %
(15S,6T,90D)	1h	6.0 %	57m 13s	0 %
(15S,8T,90D)	1h	36.1 %	1h	1.28 %
(20S,8T,90D)	1h	18.0 %	1h	0.34 %
(25S,10T,90D)	1h	61.8 %	1h	0.50 %

a trade is given by Equation 6.1, and the following modifications were done to model that a fixed number of voyages has to be serviced:

- Constraint 5.8 is removed as no voyages can be skipped.
- Constraint 5.7 is turned into the following equality constraint, stating that every voyage has to be serviced exactly once:

$$\sum_{v \in V} \sum_{k \in K_v} x_{o(v)rik} + \sum_{v \in V} \sum_{p \in R_v} \sum_{j \in N_p} \sum_{k \in K_v} x_{vpjrik} + s_{ri} = 1, r \in R, i \in N_r \tag{6.8}$$

To make the model for fixed number of voyages as similar as possible and comparable to the full model, no further changes are made. If fixing number of voyages and choosing to disregard the possibility to accept additional spot cargo on a voyage all constraints regarding quantity transported on a voyage can be removed. This implies removing the capacity constraints 5.4 and 5.5, the demand constraint 5.6, as well as the inventory constraints 5.9-5.16 and the spot market constraint 5.22. This is however a substantial change in the model and makes it impossible to compare the objection function values or the computational times in a meaningful way.

The two instances *(4S,3T,90D)* and *(10S,4T,90D)* are solved in 18.4s and 52m 13s respectively. For all other instances, Table 19 gives the optimal solution at one hour. The results show that flexibility to skip voyages if demand is already satisfied has a positive effect on the objective function. The gap between optimal solution and best bound is however much smaller when fixing number of voyages and a more detailed analysis of the results shows that the optimal solution is determined faster when the number of voyages is fixed. This is also reasonable since less choices results in smaller branch-and-bound trees, fewer computations and faster determination of the optimal solution. For the two largest instances, the model where voyages are fixed actually is able to determine a better objective function value. Note however, that the gap for these instances is much tighter when voyages are fixed. The objective value is also higher than the best bound

Table 19: Objective function values (USD) when fixing number of voyages.

	Full model		**Fixed voyages**		**Difference**	
Instance	**Obj.val.**	**Gap**	**Obj.val.**	**Gap**	**Obj.val.**	**Gap**
(4S,3T,90D)	6 541 743	0.0 %	6 683 073	0 %	2.2 %	0 %
(8S,3T,90D)	6 597 846	13.8 %	7 357 207	10.1 %	11.5 %	-16.6 %
(10S,4T,90D)	6 577 028	7.4 %	7 265 656	0 %	10.5 %	42.0 %
(12S,5T,90D)	7 933 607	8.0 %	8 945 085	1.9 %	12.7 %	58.9 %
(15S,6T,90D)	10 993 465	6.0 %	12 025 165	4.5 %	9.4 %	56.8 %
(15S,8T,90D)	14 268 801	36.1 %	12 143 567	15.7 %	-14.9 %	-141.3 %
(20S,8T,90D)	15 866 013	18.0 %	14 492 726	2.6 %	-8.7 %	-148.2 %

Table 20: Number of voyages performed.

Instance	**Voyages performed**	**Voyages on trade**
(4S,3T,90D)	11	12
(8S,3T,90D)	16	19
(10S,4T,90D)	18	21
(12S,5T,90D)	24	27
(15S,6T,90D)	29	34
(15S,8T,90D)	30	33
(20S,8T,90D)	41	42
(25S,10T,90D)	51	51

determined for the full model, with 12 143 567 exceeding 9 119 089 and 14 492 726 exceeding 13 015 127, for instance *(15S,8T,90D)* and *(20S,8T,90D)* respectively. This is the same effect as identified when large instances are solved heuristically. The results obtained when allowing longer solution time confirms the results in the table by showing that after 11h 45m, the optimal solution to *(20S,8T,90D)* had an objective function value of 14 005 143, which is less than the corresponding value when fixing the number of voyages.

Table 20 gives an overview over how many voyages that are performed when solving the full model. The column, *Voyages on trade* states how many voyages that are necessary if no large ships are used, and thus the number of voyages performed when fixing number of voyages.

6.4.3 Fixed speed

More choices imply more flexibility, a larger solution space and better solutions. Offering the possibility to reduce costs by reducing operational speed on ships when time allows it should improve the value of the objective function. Choices do however imply more variables, and more variables increase computational time. This test investigates

Table 21: The ratio of "low" speed voyages.

Instance	Voyages performed at "low" speed	in total	Share of "low" speed voyages
(4S,3T,90D)	4	11	36 %
(8S,3T,90D)	12	16	75 %
(10S,4T,90D)	16	18	89 %
(12S,5T,90D)	24	24	100 %
(15S,6T,90D)	28	29	97 %
(15S,8T,90D)	24	30	80 %
(20S,8T,90D)	35	41	85 %
(25S,10T,90D)	50	51	98 %

Table 22: Objective function values (USD) when fixing speed.

Instance	Full model		Fixed speed		Difference	
	Obj.val.	Gap	Obj.val.	Gap	Obj.val.	Gap
(4S,3T,90D)	6 541 743	0 %	7 052 889	0 %	7,8 %	0.0%
(8S,3T,90D)	6 597 846	13,8 %	7 167 584	10,5 %	8,6 %	-23.8%
(10S,4T,90D)	6 577 028	7,4 %	7 551 357	4,4 %	14,8 %	-40.6%
(12S,5T,90D)	7 933 607	8,0 %	8 881 857	3,2 %	12,0 %	-59.9%
(15S,6T,90D)	10 993 465	6,0 %	11 605 811	7,8 %	5,6 %	30.9%
(15S,8T,90D)	14 268 801	36,1 %	12 705 319	15,9 %	-11,0 %	-55.9%
(20S,8T,90D)	15 866 013	18,0 %	16 283 410	16,1 %	2,6 %	-10.2%
(25S,10T,90D)	37 252 542	61,8 %	21 561 721	24,9 %	-42,1 %	-59.7%

the influence of the possibility to choose operational speed on total costs and how computational time is affected by fixing operational speed.

The choice of speed is limited to a choice from a discrete set of two speeds. "High" speed is 20 knots and "low" speed is 14 knots. When speed is fixed, the ships are restricted to operating on "high" speed only. All input values for the fixed speed case are identical to the values of "high" speed for the full model.

Table 21 shows the proportion of voyages that are serviced at "low" speed when the planners have the option to reduce speed. The numbers show that "high" speed is only preferred for a small share of the voyages. By removing the choice between "high" speed and "low" speed, and only allowing "high" speed, the results in Table 22 occured. We can in general see an increase in the costs when all voyages are enforced to be serviced at "high" speed. The exceptions are *(15,8T,90D)* and *(25S,10T,90D)*, where the optimal solutions actually are better when the speed is fixed. This is due to the same effect as seen for the heuristic approaches and when fixing number of voyages, as the solution space when fixing the speed is reduced, and thus good solutions are found

Table 23: Shared inventories.

Instance	Number of inventories with no sharing	Number of inventories when shared
(4S, 3T, 90D)	3	2
(8S, 3T, 90D)	3	3
(10S, 4T, 90D)	4	3
(12S, 5T, 90D)	5	3
(15S, 6T, 90D)	6	4
(15S, 8T, 90D)	8	6
(20S, 8T, 90D)	8	5
(25S, 10T, 90D)	10	6

and determined good faster. The results support this by showing that the gap decreses and by analyzing the best bounds for the two instances where optimal solution improves when fixing speed we see that the best bounds support the statement. Fixing speed may produce lower objective function values in one hour since more of the enumeration tree is explored, but the best bounds show that solutions with much better objective function values may exist. That it actually also does were proven when e.g. the model with data for *(25S,10T,90D)* was run for 27h 33m and obtained an objective function value of 16 512 932, which is significantly lower that the corresponding value when fixing speed. Solution time was one hour for every instance, except from for *(4S,3T,90D)* which solved in 11.1s.

6.4.4 Shared inventories

All the testing is generally done with no shared inventories. The possibility is incorporated in the code for all instances, but in general every trade operates out of its own inventory. When testing the effect of shared inventories, we have assumed that all trades with starting points in the same geographical region share one inventory. For the instance with the actual problem size, *(25S,10T,90D)*, three trades load at the US West Coast and share one inventory. For the rest of the instances, only pairs of trades share inventories. What is different between the tests are the sets in the input which allocate trades to inventories and the corresponding inventory data which is scaled up accordingly to the joint inventories. Table 23 shows number of inventories when every trades operates from its own inventory (equals number of trades) and number of inventories when trades from the same region share inventories. The results from sharing inventories are presented in Table 24. We see that solution times were approximately equal between the full model and the model where trades operating from same region shared inventory. From the objective function values we see an increase for most instances. This can only be explained by that centralized inventories increase flexibility, and lead to increased solution time. Increased flexibility eventually produces at least as good objective function values, but as seen in many tests before, larger solution space may lead to worse solution

Table 24: Objective function values (USD) when sharing inventories.

	Full model		**Shared inventory**		**Difference**	
Instance	**Obj.val.**	**Gap**	**Obj.val.**	**Gap**	**Obj.val.**	**Gap**
(4S,3T,90D)	6 541 743	0 %	6 620 674	0 %	7.8 %	0.0 %
(8S,3T,90D)	6 597 846	13.8 %	6 597 846	13.8 %	8.6 %	-0.1 %
(10S,4T,90D)	6 577 028	7.4 %	6 797 335	8.8 %	14.8 %	19.3 %
(12S,5T,90D)	7 933 607	8.0 %	7 799 038	7.1 %	12.0 %	-11.6 %
(15S,6T,90D)	10 993 465	6.0 %	11 229 765	8.4 %	5.6 %	40.6 %
(15S,8T,90D)	14 268 801	36.1 %	15 139 686	41.2 %	-11.0 %	14.1 %
(20S,8T,90D)	15 866 013	18.0 %	18 374 105	30.5 %	2.6 %	69.8 %
(25S,10T,90D)	37 252 542	61.8 %	31 806 530	55.7 %	-42.1 %	-9.9 %

values as it takes more time to detect the good solutions. It seems that the flexibility is exploited for instance *(15S,8T,90D)* and *(25S,10T,90D)*, that offer a reduction in number of inventories of two and four, respectively. It is also worth noting that *(8S,3T,90D)* produces the exact same result in both cases, which is in accordance with Table 23 that shows that no trades share inventories for this particular instance.

7 Concluding remarks

This thesis adds an exact formulation to the relatively limited work in the current literature on exact methods for the fleet deployment problem. The mathematical formulation presented shows positive results in terms of detcting good solutions that satisfies the requirements defined by the CoA without compromising flexibility.

Testing the full model on the full set of test instances with various sizes, shows that the major concern is determining how good the optimal solutions obtained are. When the instances reaches a certain size, the gap between best solution found and best bound obtained becomes significant. Analyses of the results indicate however, that a good solution is determined relatively fast in most cases. It is the determination of whether the solution is optimal that is computationally hard and time consuming. One possible explanation is the symmetric properties of the problem. An integer linear program is symmetric if its variables can be permuted without changing the structure of the problem. The fleet deployment problem studied in this thesis has such characteristics as it consists of many ships with the identical properties. Mutually swapping the values of varibles related to these ships will produce a different solution with the exact same objective function value. When using a branching technique in the solution method, symmetry leads to an enumeration tree with many isomorphic subproblems, that is subproblems with the same structural properties. Furthermore, this means that even if the optimal solution is found, the branch-and-bound tree can still get very large until optimality is proven. The results support this, as even relatively modestly sized problems are difficult to solve to optimality.

With knowledge of the exact model's challenges regarding optimality for large problem instances, the performance of heuristic approaches based on the idea behind fix-and-relax heuristics was tested. One challenge when evaluating the performance is that the exact method was unable to produce dual bounds for larger instances. Conclusions can nevertheless be drawn. The results show clearly that fixing relatively few variables in the first iteration improves solution time significantly. No approach needed more than a couple of minutes to return its optimal solution. The quality of the objective function values shows however that it is necessary to be very careful when fixing variables from the first iteration, not to compromise the best solutions. The radical improvements in solution times can be explained by the characteristics of symmetry identified. Fixing variables has thus proven itself an efficient approach to reduce symmetry. What proved itself most effective for this type of fleet deployment problem varied between the instances. Relaxing the integer constraints on all variables except from the variables concerning the one, two or three first voyages resulted in very short solution time. Enforcing binary constraints to the first voyage on half the trades returned in general better solutions, still within ten minutes. The conclusion is that a portfolio of heuristics where the number of variables to restrict, fix and relax for each iteration is varied, provides an efficient approach that in most cases will return a near optimal solution.

One focus when developing the model in this thesis was to model the real world problem as realistic as possible. No time windows were enforced, the number of voyages performed on a trade was detemined in the solution process, and speed was eligible from a discrete set. The comparative testing shows that the flexibility provided by these features led to positive results.

To keep track of cargo aboard the ships and the cargo level in the onshore inventories at all time, provided significantly better objective function values than using time windows. Inventory control proved itself superior in terms of solution flexibility and adds value to the solution by including information on available cargo at all times.

Allowing the planner to choose how many voyages to perform on a trade, by assigning ships with a specific cargo capacity to specific trades, proved itself valuable. The results show that the feature is utilized by showing that the optimal solution includes less voyages than offered in the input data. Comparing the results shows a decrease in the objective function value of 5.0% in average, when using the full model where the number of voyages is unspecified. Fixing the number of voyages has an advantage with respect to solution time, but if time is not of vital importance, the flexibility to choose voyages is a property worth investigating in further studies.

The results from testing the effect of optional speed are also positive. Compared to when speed is fixed, offering the choice between two different operational speeds reduced the objective function value with 9.5%. The specific set of operational speeds provided is decisive for the outcome, but the conclusion is that being able to vary speed through the planning period leads to improved solutions.

Summarizing the results obtained from comparing the model with more traditional approaches, we see that every feature in the model has positive effects on the optimal solution.

The trend in the literature is to describe more industrial cases and increasingly complex models [Andersson et al., 2010]. The trend also includes to formulate richer models that provides descriptions closer to real world problems. This thesis submits to this trend by offering a customized model incorporating the actual demands given in the CoA, without introducing artificial constraints that compromise solution flexibility. Fairly evenly spread voyages is a relatively new constraint in modeling the fleet deployment problem, first presented in Arnulf and Bjørkli [2010], in accordance to the authors. This is the first model that uses inventory control as a tool to model this issue. Inventoy control also enriches the model. The positive computational results give room for optimism regarding use of these ideas in optimization models in the industry. Hopefully optimization-based DSSs based on mathematical models similar to this, will be an increasingly important tool for planners solving fleet deployment problems in Saga Forest Carriers or other liner shipping companies with heterogeneous fleets in the future.

References

H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen. Industrial aspects and literature survey: Combined inventory management and routing. *Computers and Operations Research*, 37(9):1515 – 1536, 2010.

L. H. Appelgren. A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3(1):53–68, 1969.

L. H. Appelgren. Integer programming methods for a vessel scheduling problem. *Transportation Science*, 5(1):64–78, 1971.

H. S. Arnulf and A. Bjørkli. Fleet deployment in liner shipping. Master's thesis, Norwegian University of Science and Technology, 2010.

H. Benford. A simple approach to fleet deployment. *Maritime Policy and Management*, 8(4):223–228, 1981.

P. Beraldi, G. Ghiani, A. Grieco, and E. Guerriero. Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers and Operations Research*, 35(11):3644–3656, 2008.

L. Bertazzi, M. Savelsbergh, and M. G. Speranza. Inventory routing. In B. Golden, S. Raghavan, E. Wasil, R. Sharda, and S. Voß, editors, *The vehicle routing problem: latest advances and new challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 49–72. Springer US, 2008.

L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10(2):63–211, 1983.

T. B. Boffey, E. D. Edmond, A. I. Hinxman, and C. J. Pursglove. Two approaches to scheduling container ships with an application to the north atlantic route. *Journal of the Operational Research Society*, 30(5):413–425, 1979.

S. P. Bradley, A. C. Hax, and T. L. Magnanti. Applied mathematical programming. *Addison Wesley, Reading, MA*, 1977.

M. Chajakis. Reflections on a marine vessel affair. *OR/MS Today*, 26(4):32–39, 1999.

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

M. Christiansen and K. Fagerholt. Robust ship scheduling with multiple time windows. *Naval Research Logistics (NRL)*, 49(6):611–625, 2002.

M. Christiansen and B. Nygreen. A method for solving ship routing problems with inventory constraints. *Annals of Operations Research*, 81:357–378, 1998a.

M. Christiansen and B. Nygreen. Modelling path flows for a combined ship routing and inventory management problem. *Annals of Operations Research*, 82:391–413, 1998b.

M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Chapter 4: Maritime transportation. In Cynthia Barnhart and Gilbert Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 189 – 284. Elsevier, 2007.

J.F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Handbooks in operations research and management science*, volume 14, pages 367–428. Elsevier, 2007.

G. B. Dantzig and D. R. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, 1(3):217–222, 1954.

S. de Araujo, M. Arenales, and A. Clark. Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups. *Journal of Heuristics*, 13 (4):337–358, 2007.

M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

C. Dillenberger, L. F. Escudero, A. Wollensak, and W. Zhang. On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, 75(2):275–286, 1994.

J. Everett, A. Hax, V. Lewison, and D. Nutts. Optimization of a fleet of large tankers and bulkers: A linear programming approach. *Marine Technology*, pages 430–438, October 1972.

K. Fagerholt. Optimal fleet design in a ship routing problem. *International Transactions in Operational Research*, 6(5):453–464, 1999.

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

K. Fagerholt. Designing optimal routes in a liner shipping problem. *Maritime Policy and Management*, 31(4):259–268, 2004.

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

K. Fagerholt and H. Lindstad. Turborouter: An interactive optimisation-based decision support system for ship routing and scheduling. *Maritime Economics and Logistics*, 9(3):214–233, 2007.

K. Fagerholt, T. A. V. Johnsen, and H. Lindstad. Fleet deployment in liner shipping: A case study. *Maritime Policy and Management*, 36(5):397–409, 2009.

D. Ferreira, R. Morabito, and S. Rangel. Solution approaches for the soft drink integrated production lot sizing and scheduling problem. *European Journal of Operational Research*, 196(2):697–706, 2009.

S. Gelareh and D. Pisinger. Simultaneous fleet deployment and network design of liner shipping. Technical report, DTU Management Kgs. Lyngby, 2010.

D. I. Jaramillo and A. N. Perakis. Fleet deployment optimization for liner shipping part 2: Implementation and results. *Maritime Policy and Management*, 18(4):235–262, 1991.

G. Laporte and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262, 1995.

S.A. Lawrence. *International Sea Transport: The Years Ahead.* Lexington Books, Lexington, MA, 1972.

V.D. Levy, S.P. Lvov, and S.E. Lovetsky. Man-machine system for merchant fleet operation scheduling. In *Proceedings, 7th International Symposium on Transportation and Traffic Theory, Kyoto*, 1977.

M. Mohammadi, S. Fatemi Ghomi, B. Karimi, and S. Torabi. Rolling-horizon and fix-and-relax heuristics for the multi-product multi-level capacitated lotsizing problem with sequence-dependent setups. *Journal of Intelligent Manufacturing*, 21(4):501–510, 2010.

N. A. Papadakis and A. N. Perakis. A nonlinear approach to the multiorigin, multidestination fleet deployment problem. *Naval Research Logistics (NRL)*, 36(4): 515–528, 1989.

A. N. Perakis. A second look at fleet deployment. *Maritime Policy and Management*, 12(3):209–214, 1985.

A. N. Perakis. Fleet operations optimization and fleet deployment. *The Handbook of Maritime Economics and Business, LLP*, (Chapter 26):580–597, 2002.

A. N. Perakis and D. I. Jaramillo. Fleet deployment optimization for liner shipping part 1: Background, problem formulation and solution approaches. *Maritime Policy and Management*, 18(3):183–200, 1991.

A. N. Perakis and N. A. Papadakis. Fleet deployment optimization models: Part 1. *Maritime Policy and Management*, 14(2):127–144, 1987a.

A. N. Perakis and N. A. Papadakis. Fleet deployment optimization models: Part 2. *Maritime Policy and Management*, 14(2):145–155, 1987b.

B. J. Powell and A. N. Perakis. Fleet deployment optimization for liner shipping: An integer programming model. *Maritime Policy and Management*, 24(2):183–192, 1997.

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

D. Ronen. Ship scheduling: The last decade. *European Journal of Operational Research*, 71(3):325–333, 1993.

K. Uggen, M. Fodstad, and V. S. Nørstebø. Using and extending fix-and-relax to solve maritime inventory routing problems. *TOP*, pages 1–23, 2011.

UNCTAD. Review of maritime transport. Technical report, United Nations Conference on Trade and Development, Geneva, 2009.

UNCTAD. Review of maritime transport. Technical report, United Nations Conference on Trade and Development, Geneva, 2011.

www.sagafc.com. *Saga Forest Carriers*. May 2012 .

A Mathematical formulation

Sets:

V	Set of available ships.
R	Set of trades.
N_r	Set of voyages for trade $r, N_r = \{1, 2, ..., n_r\}$.
R_v	Set of trades that can be serviced by ship v.
V_r	Set of ships that can service trade r.
A_v	Set of arcs that can be sailed by ship v.
M	Set of inventories.
R_r	Set of trades that operate from the same inventory as trade r.
K_v	Set of possible speeds for ship v.

Indices:

v	ship, $v \in V$.
r, p	trade, $r, p \in R$.
i, j	voyage, $i, j \in N_r$.
$o(v)$	initial position for ship v.
$d(v)$	final position for ship v.
m	inventory, $m \in M$.
$m(r)$	inventory for trade r.
k	speed, $k \in K_v$.

Parameters:

C_{vrk}	Cost of performing a voyage on trade r by ship v at speed k.
C_r^S	Cost of performing a voyage on trade r by a spot ship.
T_{vrk}	Duration of a voyage on trade r when performed by ship v at speed k.
T_r^S	Duration of a voyage on trade r when performed by a spot ship.
$C^B_{o(v)rk}$	Ballast sailing cost from origin to first voyage on trade r at speed k.
C^B_{vrpk}	Ballast sailing cost from a voyage on trade r to a voyage on trade p at speed k.
$T^B_{o(v)rk}$	Ballast sailing duration from origin to first voyage on trade r at speed k.
T^B_{vrpk}	Ballast sailing duration from a voyage on trade r to a voyage on trade p at speed k.
D_r	Demand on trade r.
Q_v	Maximum load capacity on ship v.
Q^S	Maximum load capacity on spot ships.
G_m	Production rate at inventory m.
W_r	Minimum spread between voyages on trade r.
T	Time at the end of the planning period.
L_{MAXm}	Maximum inventory capacity at inventory m.
L_{MINm}	Minimum limit at inventory m.
L^S_{MAXr}	Maximum available spot cargo on trade r.
L_{0m}	Initial quantity at inventory m.
E_v	Earliest possible start time for ship v.
A_{ri}	Time window opens for voyage i on trade r.
B_{ri}	Time window closes for voyage i on trade r.
P_r^S	Unit revenue for additional spot cargo.

Variables:

x_{vripjk}	equals 1 if voyage i on trade r is serviced right before voyage j on trade q by ship v at speed k, and 0 otherwise.
$x_{o(v)rik}$	equals 1 if voyage i on trade r is serviced first by ship v at speed k, and 0 otherwise.
$x_{rid(v)k}$	equals 1 if voyage i on trade r is serviced last by ship v at speed k, and 0 otherwise.
$x_{o(v)d(v)}$	equals 1 if ship v does not service any voyage, and 0 otherwise.
$t_{o(v)}$	Start time from origin.
t_{ri}	start time for voyage i on trade r.
q_{ri}	Quantity loaded on voyage i.
q_{ripj}	Quantity loaded on voyage (p, j) if performed before voyage (r, i) in time.
q^S_{ri}	Quantity additional spot cargo loaded on voyage i.
l_{ri}	Available inventory on trade r at the start of voyage i.
s_{ri}	equals 1 if voyage i on trade r is serviced by a spot ship.
y_{ripj}	equals 1 if voyage (r, i) is serviced after voyage (p, j) in time.

Objective function:

$$min\,(\sum_{v\in V}\sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} C^B_{o(v)rk}x_{o(v)rik} + \sum_{v\in V}\sum_{r\in R_v}\sum_{i\in N_r}\sum_{p\in R_v}\sum_{j\in N_r}\sum_{k\in K_v}\left(C^B_{vrpk}+C_{vrk}\right)x_{vripjk} + \sum_{v\in V}\sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} C_{vrk}x_{rid(v)k} + \sum_{r\in R}\sum_{i\in N_r} C^S_r s_{ri} - \sum_{r\in R}\sum_{i\in N_r} P^S_{ri}q^S_{ri}\,)$$

Constraints:

$$x_{o(v)d(v)} + \sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} x_{o(v)rik} = 1, \qquad v\in V \quad \text{(A.1)}$$

$$x_{o(v)d(v)} + \sum_{r\in R_v}\sum_{i\in N_r}\sum_{k\in K_v} x_{rid(v)k} = 1, \qquad v\in V \quad \text{(A.2)}$$

$$\sum_{k\in K_v} x_{o(v)rik} + \sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjrik} = \sum_{k\in K_v} x_{rid(v)k} + \sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vripjk}, \qquad v\in V, r\in R_v, i\in N_r \quad \text{(A.3)}$$

$$q_{ri} + q^S_{ri} \le \sum_{v\in V} Q_v\left(\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vripjk} + \sum_{k\in K_v} x_{rid(v)k}\right) + Q^S s_{ri}, \qquad r\in R, i\in N_r \quad \text{(A.4)}$$

$$q_{ri} \le l_{ri}, \qquad r\in R, i\in N_r \quad \text{(A.5)}$$

$$\sum_{i\in N_r} q_{ri} = D_r, \qquad r\in R \quad \text{(A.6)}$$

$$\sum_{v\in V}\sum_{k\in K_v} x_{o(v)rik} + \sum_{v\in V}\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjrik} + s_{ri} \le 1, \qquad r\in R, i\in N_r \quad \text{(A.7)}$$

$$\sum_{v\in V}\sum_{k\in K_v} x_{o(v)r(i+1)k} + \sum_{v\in V}\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjr(i+1)k} + s_{r(i+1)} \le \sum_{v\in V}\sum_{k\in K_v} x_{o(v)rik} + \sum_{v\in V}\sum_{p\in R_v}\sum_{j\in N_p}\sum_{k\in K_v} x_{vpjrik} + s_{ri}, r\in R, i\in\{1,2,...,(n_r-1)\} \quad \text{(A.8)}$$

$$l_{ri} = L_{0m(r)} + G_{m(r)}t_{ri} - \sum_{j=1}^{i-1} q_{rj} - \sum_{p\in R_r}\sum_{j=1}^{n_p} q_{ripj}, \qquad r\in R, i\in N_r \quad \text{(A.9)}$$

$$l_{ri} \le L_{MAXm(r)}, \qquad r\in R, i\in N_r \quad \text{(A.10)}$$

$$l_{ri} - q_{ri} \geq L_{MINm(r)}, \qquad r \in R, i \in N_r \quad \text{(A.11)}$$
$$q_{ripj} \leq q_{pj}, \qquad (r,i),(p,j) \in A_v \quad \text{(A.12)}$$
$$q_{ripj} \leq \max_{v \in V}\{Q_v\} y_{ripj}, \qquad (r,i),(p,j) \in A_v \quad \text{(A.13)}$$
$$q_{ripj} \geq q_{pj} - \max_{v \in V}\{Q_v\}\left(1 - y_{ripj}\right), \qquad (r,i),(p,j) \in A_v \quad \text{(A.14)}$$

$$y_{ripj}\left(t_{ri} - t_{pj}\right) \geq 0 \qquad (r,i),(p,j) \in A_v \quad \text{(A.15)}$$
$$\left(1 - y_{ripj}\right)\left(t_{pj} - t_{ri}\right) \geq 0, \qquad (r,i),(p,j) \in A_v \quad \text{(A.16)}$$

$$t_{ri} \geq t_{r(i-1)} + W_r, \qquad r \in R, i \in \{2,3,...,n_r\} \quad \text{(A.17)}$$
$$\sum_{k \in K_v} x_{o(v)rik}(t_{o(v)} + T^B_{o(v)rk}) \leq \sum_{k \in K_v} x_{o(v)rik} t_{ri}, \qquad v \in V, r \in R_v, i \in N_r \quad \text{(A.18)}$$
$$\sum_{k \in K_v} x_{vripjk}\left(t_{ri} + T_{vrk} + T^B_{vrpk}\right) \leq \sum_{k \in K_v} x_{vripjk} t_{pj}, \qquad v \in V, (r,i),(p,j) \in A_v \quad \text{(A.19)}$$
$$t_{o(v)} \geq E_v, \qquad v \in V \quad \text{(A.20)}$$
$$t_{ri} \leq T, \qquad r \in R, i \in N_r \quad \text{(A.21)}$$
$$q^S_{ri} \leq L^S_{MAXr}, \qquad r \in R, i \in N_r \quad \text{(A.22)}$$

$$x_{vripjk} \in \{0,1\}, \qquad v \in V, (r,i),(p,j) \in A_v, k \in K_v \quad \text{(A.23)}$$
$$x_{o(v)rik} \in \{0,1\}, \qquad v \in V, r \in R_v, i \in N_r, k \in K_v \quad \text{(A.24)}$$
$$x_{rid(v)k} \in \{0,1\}, \qquad v \in V, r \in R_v, i \in N_r, k \in K_v \quad \text{(A.25)}$$
$$x_{o(v)d(v)} \in \{0,1\}, \qquad v \in V \quad \text{(A.26)}$$
$$y_{ripj} \in \{0,1\}, \qquad (r,i),(p,j) \in A_v \quad \text{(A.27)}$$

Constraint (A.1)-(A.3) are flow constraints. Constraint (A.1) ensures that every ship leaves its start position and constraint (A.2) that every ship reaches its destination. Constraint (A.3) ensures that every ship that reaches a node also leaves it. Constraint (A.4) ensures that the load on a voyage does not exceed the capacity of the ship, while constraint (A.5) states that the maximum load also is limited by the current goods available in stock. Constraint (A.6) states that the contractual demand needs to be satisfied.

That one voyage is serviced one time maximum is ensured by (A.7), while (A.8) ensures that if voyage(s) are skipped on a trade, it is the last one(s).

(A.9)-(A.16) regards stock available and limits for each stock. The stock available before a voyage is given by (A.9). (A.10) and (A.11) set the maximum and the minimum limits for a stock, respectively.

(A.12)-(A.16) set the value of q_{ripj} equal to q_{ri} if (r,i) is performed after (p,j) and to zero otherwise.

(A.17)-(A.21) can all be referred to as time constraints. Constraint (A.17) ensures both the spread and the order of the voyages within a trade. (A.17) ensures that every voyage is fairly evenly spread by imposing a minimum spread in time, and also that the first voyage is serviced before voyage number two and so on. (A.18) states that a ship cannot start its first voyage before reaching it from its starting position. That a ship does not start its next journey before the previous is completed is ensured by (A.19), while (A.20) ensures that a ship does not leave its start position before its available. (A.21) restricts the voyages to start within the planning period.

(A.22) ensures that the spot cargo accepted does not exceed the limit given for spot cargo.

Constraint (A.23)-(A.27) are the binary constraints.

B Mosel code

```
(!
   Implementation of mathematical formulation. The two constraints added
   to the model when introducing time windows are specified in the code.
   The other modifications done in order to facilitate the comparative
   tests are only described in the text.
!)

model superModel

options noimplicit

uses "mmxprs" !gain access to the Xpress-Optimizer solver
uses "mmodbc" !to be able to write to Excel files
uses "mmsystem" !to be able to monitor solution time

parameters
   DataFile = "input_25S_10T_90D.txt" !input file
   ResultsXLSX = "mmodbc.excel:noindex;result_25S_10T_90D.xlsx"
   ResultsTXT = "result_25S_10T_90D.txt" !results as *.txt
end-parameters

setparam("XPRS_MAXTIME",-3600) !returns the best solution after one hour
setparam("XPRS_VERBOSE",TRUE) !writes solution process data

! INDICES and NAMES
declarations
   NumShips : integer
   Ship : set of integer
   NumTrades : integer
   Trade : set of integer
   Voyage : set of integer
   MaxNumVoyages : integer
   MaxNumTradesWithSameInventory : integer
   NumInventories : integer
   Inventory : set of integer
   TradesWithSameInventory : set of integer
   MaxCapacity : integer
   NumSpeedOptions : integer
   Speed : set of integer
end-declarations
```

```
initializations from DataFile
   NumShips
   NumTrades
   NumInventories
   NumSpeedOptions
end-initializations

Ship := 1..NumShips
Trade := 1..NumTrades
Inventory := 1..NumInventories
Speed := 1..NumSpeedOptions

declarations
   NumVoyages : array(Trade) of integer !Number of voyages on each trade
   NumTradesWithSameInventory : array(Trade) of integer
end-declarations

initializations from DataFile
   NumVoyages
   NumTradesWithSameInventory
end-initializations

forward public function findMaxNumVoyages : integer
forward public function findMaxNumTradesWithSameInventory : integer

MaxNumVoyages := findMaxNumVoyages !Finds largest number of voyages
Voyage := 1..MaxNumVoyages
MaxNumTradesWithSameInventory := findMaxNumTradesWithSameInventory

if(MaxNumTradesWithSameInventory>0) then
   TradesWithSameInventory := 1..MaxNumTradesWithSameInventory
   else
      TradesWithSameInventory := 0..0
end-if

finalize(Trade); finalize(Ship); finalize(Voyage); finalize(Inventory);
finalize(TradesWithSameInventory)

declarations
   ShipNames : array(Ship) of string
   TradeNames : array(Trade) of string
end-declarations
```

```
initializations from DataFile
   ShipNames
   TradeNames
end-initializations

! CONSTANTS and SETS
declarations
!Defines which Ships vv that are compatible with which trade routes rr:
Rv : array(Ship,Trade) of integer
!Cost of performing voyage ii on trade rr by Ship vv
VoyageCost : array(Ship, Trade, Speed) of integer
!Cost of performing a voyage on trade rr by a spot ship
SpotVoyageCost : array(Trade) of integer
!Time for performing voyage ii on trade rr by Ship vv (T_riv)
VoyageTime : array(Ship, Trade, Speed) of real
!Time for performing a voyage on trade rr by a spot ship
SpotVoyageTime : array(Trade) of real
!Cost of sailing ballast from initial position to trade rr for ship vv:
BallastCostStart : array(Ship, Trade, Speed) of integer
!Cost of sailing ballast between voyages:
BallastCost : array(Ship, Trade, Trade, Speed) of integer
!Time for sailing ballast from initial position to trade rr for ship vv:
BallastTimeStart : array(Ship, Trade, Speed) of real
!Time for sailing ballast between voyages:
BallastTime : array(Ship, Trade, Trade, Speed) of real
!Earliest possible start time for Ship vv:
ShipStartTime : array(Ship) of real
!Production rate at trade rr
ProductionRate : array(Inventory) of real
!Demand on trade rr
Demand : array(Trade) of integer
!Maximum load capacity on Ship vv
ShipCapacity : array(Ship) of integer
!Maximum load capacity on spot ships
SpotShipCapacity : integer
!Minimum spread between voyages on trade rr
Spread : array(Trade) of integer
!Maximum Inventory capacity on trade rr
InventoryLimit_Min : array(Inventory) of integer
!Maximum Inventory capacity on trade rr
InventoryLimit_Max : array(Inventory) of integer
!Initial Inventory on trade rr
InitialInventory : array(Inventory) of integer
```

```
!Maximum available spot cargo on trade rr
SpotCargoAvailable_Max : array(Trade) of integer
!Time at end of planning period
EndTime : integer
!Unit revenue for additional spot cargo
SpotCargoUnitRevenue : array(Trade) of real
!Inventory allocation
InventoryForTrade : array(Trade) of integer
!Trades with same Inventory
TradesWithCommonInventory : array(Trade,Trade) of integer
!SpeedOptions
SpeedOptions : array(Ship,Speed) of string
end-declarations

initializations from DataFile
   Rv
   VoyageCost
   SpotVoyageCost
   VoyageTime
   SpotVoyageTime
   BallastCostStart
   BallastCost
   BallastTimeStart
   BallastTime
   ShipStartTime
   SpotShipCapacity
   ProductionRate
   Demand
   ShipCapacity
   Spread
   InventoryLimit_Min
   InventoryLimit_Max
   InitialInventory
   SpotCargoAvailable_Max
   EndTime
   SpotCargoUnitRevenue
   InventoryForTrade
   TradesWithCommonInventory
   SpeedOptions
end-initializations

forward public function findMaxCapacity : integer
MaxCapacity := findMaxCapacity
```

```
! VARIABLES
declarations
!x_vripjk equals 1 if (rr,ii) and (pp,jj) is serviced successively
by Ship vv, and 0 otherwise
x_vripjk : dynamic array(Ship,Trade,Voyage,Trade,Voyage,Speed) of mpvar
!x_o equals 1 if (rr,ii) is serviced first by ship vv, and 0 otherwise
x_o :        dynamic array(Ship,Trade,Voyage,Speed) of mpvar
! x_d equals 1 if (rr,ii) is serviced last by ship vv, and 0 otherwise
x_d :        dynamic array(Ship,Trade,Voyage,Speed) of mpvar
!x_od equals 1 if ship vv does not service any voyage, and 0 otherwise
x_od :     dynamic array(Ship) of mpvar
!Start time from origin for ship vv
t_o :        array(Ship) of mpvar
!Start time for service on (rr,ii)
t_ri :     array(Trade,Voyage) of mpvar
!Quantity loaded on (rr,ii)
q_ri :     array(Trade,Voyage) of mpvar
!Quantity additional spot cargo loaded on (rr,ii)
qS_ri :     array(Trade,Voyage) of mpvar
!Available Inventory on trade rr at the start of voyage ii
l_ri :     array(Trade,Voyage) of mpvar
!s_ri equals 1 if (rr,ii) is serviced by a spot ship
s_ri :      dynamic array(Trade,Voyage) of mpvar
!Quantity loaded on (pp,jj) if performed before voyage (rr,ii)
q_ripj :   dynamic array(Trade,Voyage,Trade,Voyage) of mpvar
!equals 1 if voyage (rr,ii) is permormed after voyage (pp,jj) in time
y_ripj :    dynamic array(Trade,Voyage,Trade,Voyage) of mpvar
end-declarations

!Creating x_vripjk (binary):
forall(vv in Ship, rr in Trade, ii in Voyage, pp in Trade, jj in Voyage,
kk in Speed | ii <= NumVoyages(rr) and jj <= NumVoyages(pp) and not
((rr = pp) and (ii >= jj))) do
   if ((Rv(vv,rr) = 1) and (Rv(vv,pp) = 1)) then
      create(x_vripjk(vv, rr, ii, pp, jj, kk))
      x_vripjk(vv, rr, ii, pp, jj, kk) is_binary
   end-if
end-do
```

```
!Creating x_o and x_d (binary)
forall(vv in Ship, rr in Trade, ii in Voyage, kk in Speed |
ii <= NumVoyages(rr)) do
   if ((Rv(vv,rr) = 1)) then
      create(x_o(vv, rr, ii, kk))
      create(x_d(vv, rr, ii, kk))
      x_o(vv, rr, ii, kk) is_binary
      x_d(vv, rr, ii, kk) is_binary
   end-if
end-do

!Creating x_od (binary)
forall(vv in Ship) do
   create(x_od(vv))
   x_od(vv) is_binary
end-do

!Creating s_ri (binary):
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   create(s_ri(rr,ii))
   s_ri(rr,ii) is_binary
end-do

!Creating y_riqj (binary):
forall(rr in Trade, ii in Voyage, pp in Trade, jj in Voyage | ii <=
NumVoyages(rr) and jj <= NumVoyages(pp) and not (rr=pp and jj=ii) ) do
      create(y_ripj(rr, ii, pp, jj))
      y_ripj(rr, ii, pp, jj) is_binary
end-do

!Creating q_riqj:
forall(rr in Trade, ii in Voyage, pp in Trade, jj in Voyage |
ii <= NumVoyages(rr) and jj <= NumVoyages(pp) ) do
      create(q_ripj(rr, ii, pp, jj))
end-do
```

```
! CONSTRAINTS
declarations
   Constraint_5.1  :  array(Ship) of linctr
   Constraint_5.2  :  array(Ship) of linctr
   Constraint_5.3  :  array(Ship,Trade,Voyage) of linctr
   Constraint_5.4  :  array(Trade,Voyage) of linctr
   Constraint_5.5  :  array(Trade,Voyage) of linctr
   Constraint_5.6  :  array(Trade) of linctr
   Constraint_5.7  :  array(Trade,Voyage) of linctr
   Constraint_5.8  :  array(Trade,Voyage) of linctr
   Constraint_5.9  :  array(Trade,Voyage) of linctr
   Constraint_5.10 :  array(Trade,Voyage) of linctr
   Constraint_5.11 :  array(Trade,Voyage) of linctr
   Constraint_5.12 :  array(Trade,Voyage) of linctr
   Constraint_5.13 :  array(Trade,Voyage) of linctr
   Constraint_5.14 :  array(Ship,Trade,Voyage,Speed) of linctr
   Constraint_5.15 :  array(Ship,Trade,Voyage,Trade,Voyage,Speed) of linctr
   Constraint_5.16 :  array(Ship) of linctr
   Constraint_5.17 :  array(Trade,Voyage) of linctr
   Constraint_5.18 :  array(Trade,Voyage,Trade,Voyage) of linctr
   Constraint_5.19 :  array(Trade,Voyage,Trade,Voyage) of linctr
   Constraint_5.20 :  array(Trade,Voyage,Trade,Voyage) of linctr
   Constraint_5.21 :  array(Trade,Voyage,Trade,Voyage) of linctr
   Constraint_5.22 :  array(Trade,Voyage,Trade,Voyage) of linctr
end-declarations

!Flow constraints:
!Every ship is required to leave its start position:
forall(vv in Ship) do
   Constraint_5.1(vv) :=
      x_od(vv) + sum(rr in Trade, ii in Voyage, kk in Speed |
      ii<=NumVoyages(rr)) x_o(vv,rr,ii,kk) = 1
end-do

!Every ship is required to reach its destination node:
forall(vv in Ship) do
   Constraint_5.2(vv) :=
      x_od(vv) + sum(rr in Trade, ii in Voyage, kk in Speed |
      ii<=NumVoyages(rr)) x_d(vv,rr,ii,kk) = 1
end-do
```

```
!Node balance for every voyage node:
forall(vv in Ship, rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.3(vv,rr,ii) :=
      sum(kk in Speed) x_o(vv,rr,ii,kk) + sum(pp in Trade, jj in Voyage,
      kk in Speed | jj<=NumVoyages(pp))(x_vripjk(vv,pp,jj,rr,ii,kk)) =
      sum(kk in Speed) x_d(vv,rr,ii,kk) + sum(pp in Trade, jj in Voyage,
      kk in Speed | jj<=NumVoyages(pp))(x_vripjk(vv,rr,ii,pp,jj,kk))
end-do

!Quantity constraints:
!A ship can not accept more cargo than its capacity:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.4(rr,ii) :=
      q_ri(rr,ii)+qS_ri(rr,ii) <= sum(vv in Ship)(ShipCapacity(vv)*
      (sum(pp in Trade,jj in Voyage,kk in Speed | jj<=NumVoyages(pp))
      (x_vripjk(vv,rr,ii,pp,jj,kk)) + sum(kk in Speed)(x_d(vv,rr,ii,kk)))) +
      SpotShipCapacity*s_ri(rr,ii)
end-do

!A ship can not accept more cargo than currently available in inventory:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.5(rr,ii) :=
      q_ri(rr,ii) <= l_ri(rr,ii)
end-do

!The contractual demand needs to be satisfied:
forall(rr in Trade) do
   Constraint_5.6(rr) :=
      sum(ii in Voyage | ii<=NumVoyages(rr))(q_ri(rr,ii)) = Demand(rr)
end-do

!Voyage completion constraints:
!One voyage can be serviced one time maximum:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.7(rr,ii) :=
      sum(vv in Ship, kk in Speed)(x_o(vv,rr,ii,kk)) +
      sum(vv in Ship, pp in Trade, jj in Voyage, kk in Speed |
      jj<=NumVoyages(pp)) (x_vripjk(vv,pp,jj,rr,ii,kk)) + s_ri(rr,ii)<= 1
end-do
```

```
!To ensure that if voyage(s) are skipped on a trade, it is the last one(s):
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)-1) do
   Constraint_5.8(rr,ii) :=
      sum(vv in Ship, kk in Speed)(x_o(vv,rr,ii,kk)) +
      sum(vv in Ship, pp in Trade, jj in Voyage, kk in Speed |
      jj<=NumVoyages(pp))(x_vripjk(vv,pp,jj,rr,ii,kk)) + s_ri(rr,ii)>=
      sum(vv in Ship, kk in Speed)(x_o(vv,rr,ii+1,kk)) +
      sum(vv in Ship, pp in Trade, jj in Voyage, kk in Speed |
      jj<=NumVoyages(pp))(x_vripjk(vv,pp,jj,rr,ii+1,kk)) + s_ri(rr,ii+1)
end-do
!Harbor inventory constraints:
!The cargo available for a voyage:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.9(rr,ii) :=
      l_ri(rr,ii) = InitialInventory(InventoryForTrade(rr)) +
      ProductionRate(InventoryForTrade(rr))*t_ri(rr,ii) -
      sum(jj in 1..(ii-1))(q_ri(rr,jj)) -
      sum(pp in 1..NumTradesWithSameInventory(rr) |
      NumTradesWithSameInventory(pp)>0,jj in 1..NumVoyages(
      TradesWithCommonInventory(rr,pp)) | jj<=
      NumVoyages(TradesWithCommonInventory(rr,pp)))
      (q_ripj(rr,ii,TradesWithCommonInventory(rr,pp),jj))
end-do
!The cargo available must be within limits before and after each voyage:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.10(rr,ii) :=
      l_ri(rr,ii) <= InventoryLimit_Max(InventoryForTrade(rr))
end-do
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.11(rr,ii) :=
      l_ri(rr,ii) - q_ri(rr,ii) >= InventoryLimit_Min(InventoryForTrade(rr))
end-do
!q_ripj equals q_ri if (rr,ii) is performed after (pp,jj) and 0 otherwise:
forall(rr in Trade,ii in Voyage, pp in Trade, jj in Voyage |
ii<=NumVoyages(rr) and jj <= NumVoyages(pp)) do
   Constraint_5.12(rr,ii,pp,jj) :=
      q_ripj(rr,ii,pp,jj) <= q_ri(pp,jj)
end-do
forall(rr in Trade,ii in Voyage, pp in Trade, jj in Voyage |
ii<=NumVoyages(rr) and jj <= NumVoyages(pp)) do
   Constraint_5.13(rr,ii,pp,jj) :=
      q_ripj(rr,ii,pp,jj) <= MaxCapacity*y_ripj(rr,ii,pp,jj)
end-do
```

```
forall(rr in Trade,ii in Voyage, pp in Trade, jj in Voyage |
ii<=NumVoyages(rr) and jj <= NumVoyages(pp)) do
   Constraint_5.14(rr,ii,pp,jj) :=
      q_ripj(rr,ii,pp,jj) >= q_ri(pp,jj) - MaxCapacity*(1-y_ripj(rr,ii,pp,jj))
end-do
!Setting y_ripj to 1 if (rr,ii) is performed after (pp,jj) and 0 otherwise:
forall(rr in Trade,ii in Voyage, pp in Trade, jj in Voyage |
ii<=NumVoyages(rr) and jj <= NumVoyages(pp)and not (rr=pp and jj=ii) ) do
   Constraint_5.15(rr,ii,pp,jj) :=
      t_ri(rr,ii) - t_ri(pp,jj) >= EndTime*(y_ripj(rr,ii,pp,jj)-1)
end-do
forall(rr in Trade,ii in Voyage, pp in Trade, jj in Voyage |
ii<=NumVoyages(rr) and jj <= NumVoyages(pp)and not (rr=pp and jj=ii) ) do
   Constraint_5.16(rr,ii,pp,jj) :=
      t_ri(pp,jj) - t_ri(rr,ii) >= -EndTime*y_ripj(rr,ii,pp,jj)
end-do

!Time constraints:
!Voyages need to be fairly evenly spread and serviced in order:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr) and ii > 1) do
   Constraint_5.17(rr,ii) :=
      t_ri(rr,ii) >= t_ri(rr,ii-1) + Spread(rr)
end-do
!A ship cannot start its first voyage before reaching it from the start:
forall(vv in Ship, rr in Trade, ii in Voyage,
kk in Speed | ii <= NumVoyages(rr)) do
   Constraint_5.18(vv,rr,ii,kk) :=
      t_o(vv) + BallastTimeStart(vv,rr,kk) - t_ri(rr,ii) <=
      (ShipStartTime(vv) + BallastTimeStart(vv,rr,kk))*(1-x_o(vv,rr,ii,kk))
end-do
!A ship cannot start its next journey before the previous is completed:
forall(vv in Ship, rr in Trade, ii in Voyage, pp in Trade, jj in Voyage,
kk in Speed | ii <= NumVoyages(rr) and jj <= NumVoyages(pp) ) do
   Constraint_5.19(vv,rr,ii,pp,jj,kk) :=
      t_ri(rr,ii)+VoyageTime(vv,rr,kk)+BallastTime(vv,rr,pp,kk)-t_ri(pp,jj)<=
      (EndTime - ShipStartTime(vv)*(1-x_vripjk(vv,rr,ii,pp,jj,kk))
end-do
!A ship cannot leave its start position before it is available:
forall(vv in Ship) do
   Constraint_5.20(vv) :=
      t_o(vv) >= ShipStartTime(vv)
end-do
```

```
!All voyages must be completed within the planning period:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.21(rr,ii) :=
      t_ri(rr,ii) <= EndTime
end-do
!Spot market constraints:
!A ship has an upper limit on quantity of spot cargo:
forall(rr in Trade, ii in Voyage | ii <= NumVoyages(rr)) do
   Constraint_5.22(rr,ii) :=
      qS_ri(rr,ii)<=SpotCargoAvailable_Max(rr)
end-do
!*************************************************************************
!The following constraints are added when introducing time windows:
!Time window constraints:
forall(rr in Trade, ii in Voyage | ii<=NumVoyages(rr)) do
   Constraint_TW_min(rr,ii) :=
      t_ri(rr,ii)>=TW_min(rr,ii)
end-do
forall(rr in Trade, ii in Voyage | ii<=NumVoyages(rr)) do
   Constraint_TW_max(rr,ii) :=
      t_ri(rr,ii)<=TW_max(rr,ii)
end-do
!*************************************************************************
! OBJECTIVE FUNCTION
declarations
   Cost : linctr
end-declarations

Cost:=
   sum(vv in Ship,rr in Trade,ii in Voyage,kk in Speed | ii<=NumVoyages(rr))
   (BallastCostStart(vv,rr,kk)*x_o(vv,rr,ii,kk)) +
   sum(vv in Ship,rr in Trade,ii in Voyage, pp in Trade,jj in Voyage,
   kk in Speed | ii<=NumVoyages(rr) and jj<=NumVoyages(pp))
   ((BallastCost(vv,rr,pp,kk) + VoyageCost(vv,rr,kk)) *
   x_vripjk(vv,rr,ii,pp,jj,kk)) +
   sum(vv in Ship,rr in Trade,ii in Voyage,kk in Speed | ii<=NumVoyages(rr))
   (VoyageCost(vv,rr,kk) * x_d(vv,rr,ii,kk)) +
   sum(rr in Trade,ii in Voyage | ii<=NumVoyages(rr))
   (SpotVoyageCost(rr) * s_ri(rr,ii)) -
   sum(rr in Trade,ii in Voyage | ii<=NumVoyages(rr))
   (SpotCargoUnitRevenue(rr) * qS_ri(rr,ii))

minimize (Cost )
```

```
declarations
   End_time : real
end-declarations
End_time := gettime

!Write solution to file
declarations
   ObjectiveValue : real
   BestBound : real
   Gap : real
   CurrentTime : string
   CurrentDate : string
   SolutionTime : real
   TotalNumberOfVoyages : integer
end-declarations
ObjectiveValue := getobjval
BestBound := getparam('XPRS_bestbound')
if(ObjectiveValue<>0) then
   Gap := (1-BestBound/ObjectiveValue)*100 !%
   else Gap := 0
end-if
CurrentTime := getparam('parser_time')
CurrentDate := getparam('parser_date')
SolutionTime := End_time-Start_time
TotalNumberOfVoyages:=0
forall(rr in Trade) do
   TotalNumberOfVoyages += NumVoyages(rr)
end-do

initializations to ResultsXLSX
   DataFile as 'Inputfile'
   CurrentDate as 'CurrentDate'
   CurrentTime as 'CurrentTime'
   SolutionTime as 'SolutionTime'
   ObjectiveValue as 'ObjectiveValue'
   BestBound as 'BestBound'
   Gap as 'Gap'
   NumVessels as 'Vessels'
   NumTrades as 'Trades'
   NumStocks as 'Stocks'
   TotalNumberOfVoyages as 'TotalNumberOfVoyages'
   EndTime as 'EndTime'
end-initializations
```

```
fopen(ResultsTXT,F_OUTPUT)
!Write running data
writeln("Inputfile: "+DataFile)
writeln("Date and time: "+CurrentDate+" "+CurrentTime); writeln("")
!Write solutions to output
writeln("Total cost: "+ getobjval)
writeln("Best bound: "+ BestBound)
writeln("Gap: "+Gap)
writeln("Solution time: "+SolutionTime+"s")
writeln("")
writeln("End time: "+EndTime)
writeln("Total number of voyages: "+TotalNumberOfVoyages)
forall(rr in Trade) do
   writeln("Number of voyages on Trade "+rr+": "+NumVoyages(rr))
end-do

! FUNCTIONS
!Returns the largest number of voyages in any trade:
public function findMaxNumVoyages : integer
   declarations
      maxValue : integer
   end-declarations
   maxValue := 0
   forall(rr in Trade) do
      if NumVoyages(rr) > maxValue then
         maxValue := NumVoyages(rr)
      end-if
   end-do
   returned := maxValue
end-function

public function findMaxNumTradesWithSameInventory : integer
   declarations
      maxValue : integer
   end-declarations
   maxValue := 0
   forall(rr in Trade) do
      if NumTradesWithSameInventory(rr) > maxValue then
         maxValue := NumTradesWithSameInventory(rr)
      end-if
   end-do
   returned := maxValue
end-function
```

```
public function findMaxCapacity : integer
   declarations
      maxValue : integer
   end-declarations
   maxValue := 0
   forall(vv in Ship) do
      if ShipCapacity(vv) > maxValue then
         maxValue := ShipCapacity(vv)
      end-if
   end-do
   returned := maxValue
end-function

end-model
```