ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Industrial and Tramp Ship Routing and Scheduling Problems with Speed Optimization and Split Loads

Bachelor of Science (BSc) in Econometrics and Operations Research

Bachelor Thesis

Quantitative Logistics & Operations Research

Name: Lisanne van Huizen
Student Number: 427878

Supervisor: Nemanja Milovanovic
Second Assessor: Remy Spliet

July 8, 2018

Industrial and Tramp Ship Routing and Scheduling Problems with Speed Optimization and Split Loads

Abstract

Industrial and tramp shipping are two of the main shipping industries within seaborne transportation. For those industries, the goal is to minimize the total operational cost of the fleet plus the use of spot charters when not all cargoes can be transported by the fleet. Hence, efficient schedules must be made for the vessels such that at minimum cost all cargoes are transported. In the existing literature, this problem is referred to as the industrial and tramp ship routing and scheduling problem (ITSRSP). The purpose of our thesis is to analyze whether extending this problem to more real-life applications results in lower total operational cost. We focus on three extensions of the industrial and tramp ship routing and scheduling problem. Namely, to speed optimization, to split loads and to both speed optimization and split loads.

The results showed that extending the ITSRSP to speed optimization, could reduce the total cost on average with 25% compared to the cost of ITSRSP without any extensions. For the extension to split loads, the results showed that the operational cost could reduce up to around 16% compared to the cost without extensions. However unfortunately, the average cost reduction for this extension is only 2%. Lastly, the extension to both speed optimization and split loads results in an average cost reduction of 22%. This leads to the conclusion that extending the problem to speed optimization results on average in the highest cost reduction. Nevertheless, we recommend to further analyze the extensions to split loads and to speed optimization combined with split loads, such that more significant results can be computed. We expect that this would result in even more cost reductions than is made by only the extension to speed optimization.

Contents

Chapter 1

Introduction

The world economy strongly relies on trade, due to differences in manufacturing, differences in natural resources and temporary imbalances between countries. Large volumes of cargo have to be transported between different continents. For this, the only cost-effective transportation mode is seaborne transportation. UNCTAD (2017) reported that even more than ten billion tons of goods are carried at sea annually by a world fleet with a capacity of more than 1.86 billion deadweight tons.

In the existing literature, a distinguishing between three different transportation modes within seaborne transportation is commonly made. These three transportation modes are liner shipping, tramp shipping and industrial shipping. Liner shipping is mainly the shipping of containers. The vessels sail a fixed route, and the frequency of the sailing of the route is known far in advance, just like it is for bus lines. Industrial and tramp shipping show several similarities in their planning characteristics. They both primarily transport oil and gas products and dry bulk commodities. However, in the industrial shipping market, an operator owns the cargoes and controls the ships. The operator tries to minimize the transportation cost of the cargoes. While in the tramp shipping market, a company has contracts which specify the mandatory number of cargoes that the company must transport. The ships of the company usually transport both these mandatory cargoes and optional spot cargoes, to maximize profit.

UNCTAD (2017) further reported that 16.7% of all trade volume is transported by container vessels, while 47.5% of all trade volume is transported by bulk carriers and 29.7% by oil tankers. This shows that industrial and tramp shipping contribute in huge volume to the total seaborne trade. For the transportation of these volumes holds that the cargoes must be picked up at the loading port and delivered at the corresponding discharging port, within a certain time interval. Hence, given the large volumes within industrial and tramp shipping, improving efficiency in these transportation modes is of high importance as this could have a huge impact on operational cost and thereby also on the environment. The planning of the transportation between the loading and the discharging port, while minimizing total cost, is referred to as the ship routing and scheduling problem.

For the basic implementation of the ship routing and scheduling problem, usually a fixed speed is used for each vessel during the entire planning period. Furthermore, each cargo needs to be picked up and delivered by the same ship in time. In reality however, vessels are able to sail at different speeds, within the range of the minimum and maximum speed of that vessel. As the fuel consumption and thus the fuel cost is related to the speed of a vessel, optimizing speed could have a huge effect on the operational cost. Further, cost can also be reduced by allowing split loads, by which we mean allowing one cargo to be transported from origin to destination by several ships instead of one. Therefore, the goal of our research is to extend the basic industrial and tramp ship routing and scheduling problems to more real-life applications in which we introduce variable speeds during the planning period and allow for split loads.

Among others, Hemmati et al. (2014) argued that despite the high importance of industrial and tramp ship routing and scheduling problems, researchers are often reluctant to share their performed research. This is mainly because their instances do not capture all features of the maritime transportation. Therefore, Hemmati et al. (2014) made available a wide range of realistic benchmark instances for these routing and scheduling problems, as well as an instance generator for generating additional instances. The purpose of the available benchmark instances and the instance generator is that better solution algorithms can be developed and tested for the industrial and tramp ship routing and scheduling problems. Furthermore, they recommended modeling the problem to more real-life extensions. For example, they suggested extending the problem to include speed optimization, bunkering decisions and environmental regulations. This again indicates the importance of our research, as we extend the basic problem to incorporate both speed optimization and split loads.

The remainder of the paper is organized as follows. In Chapter 2, we review relevant literature on the basic industrial and tramp ship routing and scheduling problems, as well as solution methods for this problem. Also, we review literature on the same extensions we make in our research, namely speed optimization and split loads. We continue thereafter in Chapter 3 with a detailed description of the basic industrial and tramp ship routing problems and the extensions of the basic problem. Chapter 4 describes the data used to test the models and solution approaches. The solution approaches are explained in Chapter 5. Besides the mixed-integer programming formulations of both versions of the model, we propose an adaptive large neighborhood search heuristic as solution method. This method is also used by, among others, Hemmati et al. (2014) and Ropke and Pisinger (2006). As this heuristic does not incorporate split loads and speed optimization, we also discuss two necessary extension of this heuristic, in this chapter. We continue in Chapter 6 with presenting and discussing the results obtained by the solution methods. We focus on the best obtained results of each solution method for each problem, and the performance of the solution methods. We also focus on the cost improvements made by extending the problem to more real-life applications, as this is the main goal of our research. Finally, in Chapter 7 we state our concluding remarks and recommendations for further research.

Chapter 2

Literature Review

During the last decades, research in industrial and tramp ship routing and scheduling problems has tremendously increased. Christiansen et al. (2013) provided an extensive review on research on these problems and extensions thereof, which is the fourth review in a series of reviews. We base our literature review on theirs such that a correct review on research is given. They stated that since the last decade the research volume has more than doubled. However, due to the complexity of the problem, opportunities for improvements and extensions remain for future research. As the volume of research on this problem is very large, we focus only on reviewing literature which is in line with our research. We refer to Christiansen et al. (2013), Christiansen et al. (2004), Ronen (1993) and Ronen (1983) for extensive literature reviews.

In the existing literature, most research is done on the tramp ship routing and scheduling problem (TSRSP). This is due to the fact that the industrial ship routing and scheduling problem (ISRSP) is considered a special case of the TSRSP. The objective in most published work is therefore to maximize the profit obtained by transporting both mandatory and optional spot cargoes, instead of minimizing the operational cost. In these TSRSPs, the speed of each vessel and its fuel consumption is considered fixed during the entire planning period. Also, each cargo needs to be transported by one ship, meaning that split loads are not allowed.

For solving these problems, Brønmo et al. (2007) introduced a multi-start local search heuristic, of which the performance is compared to a path flow formulation where, a priori, all ship routes are generated. Note that the path flow formulation is not recommended for solving large instances, as the number of possible routes grow exponentially with the size of the problem instance. Korsvik et al. (2010) compared the performance of this multi-start local search heuristic with a tabu search. The tabu search allows validation of the capacity and time window constraints during the search to an optimal solution. Even though the tabu search outperforms the multi-start local search heuristic, Malliappi et al. (2011) introduced an even better method, namely a variable neighborhood search which outperforms both the multi-start local search heuristic and the tabu search. Further, Lin and Liu (2011) proposed a genetic algorithm to solve the problem and Brønmo et al. (2010) a heuristic column generation approach. Most of the heuristics discussed above are included in a decision support system (DSS) for ship routing and scheduling problems, presented by Korsvik and Fagerholt (2010).

Instead of only considering the tramp ship routing and scheduling problem, Hemmati et al. (2014) formulated a combined industrial and tramp ship routing and scheduling problem (ITSRSP) of which the objective is to minimize the total costs of operating the fleet plus the cost of using spot charters to transport the cargoes. Besides the mathematical formulation of the model, they also presented an adaptive large neighborhood search (ALNS) implementation to solve the problems. For a detailed description of the ALNS heuristic, they referred to Ropke and Pisinger (2006).

As discussed previously, in most of the literature on ship routing and scheduling, a cargo must be transported by one vessel. By introducing split loads, a cargo may be split among several vessels as it is transported from origin to destination. This extended problem is studied by Andersson et al. (2011) and Korsvik et al. (2011). Andersson et al. (2011) proposed a solution method based on the generation of single vessel routes a priori, and two alternative path flow models. These solution methods perform well for small instances, however as the number of routes exponentially increases with the size of the problem instance, Korsvik et al. (2011) suggested a large neighborhood search heuristic. This heuristic provides optimal solutions for both small and larger routing and scheduling problems.

Besides the extension to split loads, also the extension to speed optimization has received much attention in the last decade. This is mainly because fuel prices increased tremendously, and regulations were set regarding environmental emissions. Among others, Norstad et al. (2011) and Psaraftis and Kontovas (2013) discussed similar problems. They both presented methods for solving the problem, by considering speed as decision variable. Psaraftis and Kontovas (2013) mainly focused on reducing the effects of CO_2 emission by optimizing speed. Norstad et al. (2011) focused on minimizing cost by optimizing speed. They formulated a speed optimization problem (SOP) for a fixed ship route. For the SOP two solution methods are proposed, namely the discretization of the arrival times at the ports and the recursive smoothing algorithm (RSA). In the RSA, the average speed between ports on the route is adjusted until a feasible and optimal solution is found. They found that the RSA outperforms the discretization of the arrival times, and that the difference in their performance is more feasible as the size of the problem instance increases.

As far as we could find, up to now, the extension of the industrial and tramp ship routing and scheduling problem to both speed optimization and split loads is not yet analyzed. However, both extensions, in itself, have shown to result in lower total operational cost of the fleet in all researches. We hope with our thesis to provide more insight in the possible cost reductions of the extended variant of the industrial and tramp ship routing and scheduling problem, which includes both speed optimization and split loads.

Chapter 3

Problem Description

In this chapter we discuss the basic and extended industrial and tramp ship routing and scheduling problems in detail. As mentioned before, our focus lies on these problems as industrial and tramp shipping have similarities regarding operational characteristics and they both contribute in huge volumes to the total world trade.

Hemmati et al. (2014) stated that in industrial shipping market, an operator owns the cargoes and controls the ships of the fleet. The operator tries to minimize the transportation cost of the cargoes. In contrast to the tramp shipping market, where a company has several long-term contracts in which the mandatory number of cargoes that the company is committed to carry is specified. The ships of the company transport usually both these mandatory cargoes and, as many as possible, optional spot cargoes, to maximize profit. As the objectives of the shipping industries differ, we reformulate the tramp shipping objective to the minimization of the transportation cost such that both ship routing and scheduling problems can be solved by the same mixed-integer programming model. Note that this is also done by Hemmati et al. (2014), and we rely on their formulation of the problem.

In Section 3.1 we provide a detailed description of the basic industrial and tramp ship routing and scheduling problems (ITSRSP). We continue in Section 3.2, with a description of the extensions of the basic ITSRSP, which are the extensions to speed optimization and split loads.

3.1 Industrial and Tramp Ship Routing and Scheduling Problems

The objective of the ITSRSP is to minimize the operational cost of each vessel, in such a way that all mandatory and spot cargoes are picked up, by a vessel of the fleet or by a spot charter, in time. This means that for each vessel of the fleet a schedule must be made which specifies the order of ports it needs to visit, for minimum cost.

The fleet of vessels is owned by an operator or a company and can be homogeneous or heterogeneous. However, within industrial and tramp shipping it is more common to consider a heterogeneous fleet of vessels. In the basic ITSRSP, the speed of a vessel and its fuel consumption is considered fixed. Each vessel has a maximum capacity, and the total volume of the cargo quantities on board may not exceed this capacity. The fleet is considered fixed during the entire planning period, and therefore the fixed cost of vessels, such as the cost for maintenance, is not taken into account. For the total variable cost only port costs, sailing costs and spot charter costs (if necessary) are included. Port costs are ship dependent and they are incurred when a ship loads or unloads cargo at the port. At each port, vessels have different service times for loading and unloading, and their service at the loading or discharging ports must start within a specified time window.

For each cargo, the time windows are specified by an earliest and latest start time in which the pick-up and delivery can take place. Further, each cargo is characterized by its fixed quantity and its cost when the cargo needs to be transported by a spot charter.

3.2 Incorporating Speed Optimization and Split Loads

As mentioned previously, the basic ITSRSP assumes that the speed of each vessel is kept fixed during the entire planning period and that each cargo must be transported by only one vessel from origin to destination. As this is not very realistic, we extend the basic ITSRSP to incorporate speed optimization and split loads. The full name of this problem is the industrial and tramp ship routing and scheduling problem with speed optimization and split loads (ITSRSPSOSL).

The used fixed speed of a ship in the ITSRSP is often the design speed of a ship. However, in reality, ships can sail at different speeds within their range of the minimum and maximum speed. As the fuel cost of a ship depends on its sailing speed, optimizing the sailing speed can lower the fuel cost. Hence, we have to take speed as a decision variable for each sailing leg, in order to further reduce the total operational cost. Besides the speed of each vessel, the sailing cost of each leg for each vessel in the ITSRSP is also fixed because the distance of each sailing leg, and thus the travel time at the fixed speed, is known in advance. In the ITSRSPSOSL, we allow variable speed, which means that the travel time from port to port is no longer known in advance. Therefore, we have to determine the fuel cost per hour for each vessel as a function of speed.

Besides assuming a fixed speed and fixed sailing cost of each vessel during the entire planning period, the ITSRSP also assumes that each cargo must be transported by only one vessel from origin to destination. In other words, the ITSRSP does not allow for split loads. Hence, we again extend the formulation such that also split loads are allowed. This means that a cargo can be transported by several ships, as long as the total quantity of the cargo is picked up and delivered within the specified time windows at the loading and discharging port. It thus is important to keep track of the quantity of the cargo transported by a ship. The total quantity of cargoes on board of a ship may still not exceed the capacity of the ship. By allowing split loads, the ship utilization can improve such that within tramp shipping the opportunity arises to transport more spot cargoes and within industrial and tramp shipping the cost for outsourcing the cargoes to spot charters may reduce. In other words, the extension to split loads can reduce the total operational cost of the fleet.

Chapter 4

Data

Hemmati et al. (2014) provided an instance generator to allow other researchers to test new models and solution approaches. We use their instance generator to create a total of 18 new instances. Normally, a distinguishing is made in full ship load and mixed ship load for the type of cargo sizes. Hemmati et al. (2014) stated that, for the major bulk commodities, the cargo is usually a full ship load. Whereas for the minor bulk commodities, the cargoes are smaller and therefore ships can carry different cargoes simultaneously, which results in a mixed ship load. We only consider the mixed load case, as we want to extend the basic industrial and tramp ship routing and scheduling problem to incorporate split loads. In case of full ship loads, allowing for split loads does not change much as each ship can transport only one cargo each time. Therefore, the basic problem and the extended problem (without speed optimization) result in the almost the same problem for full ship loads, which is not very interesting.

Before we generate the instances, we form a short sea data set, in which we make a division of 16 different countries in the Mediterranean region. Our data set is based on the data sets provided by Brouer et al. (2013) and Hemmati et al. (2014). In total, 39 ports are included in the short sea port set. The location of each port is shown in Figure 4.1, by its corresponding port code. For each of these 39 ports, the region, the coordinates, the distance to all other ports and the fixed cost of berthing at each port for each vessel, is given in the data set.

Figure 4.1: The 39 ports in the Mediterranean region

The Figure shows the locations of all 39 ports, which are included in the short sea port set. Each of these ports is presented by a dot and its corresponding port code in the map of the Mediterranean region.

All 18 instances are based on a normal market condition. This means that there will be a relative average number of cargoes available to be transported, and that the cost of using spot charters is neither high nor low. Also, the instances have unbalanced flow between regions, which means that some regions are mainly exporters of goods and others mainly importers of goods. The latter is more realistic than considering balanced flow between regions. Furthermore, each instance has 'normal' time windows. The 'normal' time window for the loading port consist of three consecutive days and for the discharging port, the time window is usually wider. The latter is to allow ships to load and unload the cargoes at the loading and discharging ports, respectively.

Finally, all instances have a heterogeneous fleet of vessels. In total, six types of short sea vessels are included in the data set. For each ship, the capacity, the minimum speed, the maximum speed, the design speed and the fuel consumption at design speed are given in the data set and stated in Table 4.1.

Table 4.1: Data of vessel types

Vessel Type	Capacity (dwt)	Min. Speed (knots)	Max. Speed (knots)	Design Speed (knots)	Fuel Consumption* (MT/day)
Feeder 4500	4500	10	14	12	18.8
Feeder 8000	8000	10	17	14	23.7
Panamax 12000	12000	12	19	18	52.5
Panamax 24000	24000	12	22	16	57.4
Post Panamax	42000	12	23	16.5	82.2
Super Panamax	75000	12	22	17	126.9

* The fuel consumption (in metric ton per day) for each ship when sailing at the design speed. The Table states the capacity in deadweight tons, the minimum speed, the maximum speed and the design speed, all in knots, and the fuel consumption at design speed for each of the six different vessel types.

For each of the 18 instances, we based the input for the instance generator on the input of Hemmati et al. (2014). For each of their instances, the included vessel types and the total number of vessels for each type is stated. Using the same input, the following information is given after generating each of our instances with the instance generator provided by Hemmati et al. (2014).

For each vessel, the home port, the time it comes available at the home port and the capacity of that vessel, is given. For each cargo, the origin port, the destination port, the quantity, the costs of not transporting, and the start and end times for the time windows for the pick-up and for the delivery, are stated. Also, for each included vessel, the travel time in hours and the travel cost in USD between all ports when sailing at design speed, is given. The times and costs between the ports for each vessel are based on the given distances between the ports, fuel consumption for sailing at the design speed and the IFO fuel price of 590 USD per MT (metric ton). Finally, for each included vessel, the service time in hours for the origin port and the destination port, with corresponding port cost in USD, is given for each cargo.

Chapter 5

Solution Approaches

In this chapter, we discuss the solution approaches for solving the instances described in Chapter 4. We use different approaches for both variants of the problem. First, we focus on the basic industrial and tramp ship routing and scheduling problem, which we described in Section 3.1. Thereafter, we focus on the extended version, in which we incorporate speed optimization and split loads, as we described in Section 3.2.

We start with providing the mixed-integer programming models of both versions of the problem in Section 5.1. These mixed-integer programming models can be solved by commercial optimization software, which in fact is also be done by Hemmati et al. (2014). However, they showed that solving the mixed-integer programming model with commercial software goes well for instances up to around 18 cargoes. Of course, we also want to optimize operational cost for instances with more than 18 cargoes. Therefore, Hemmati et al. (2014) proposed the adaptive large neighborhood search (ALNS) heuristic, which is discussed in detail by Ropke and Pisinger (2006). The ALNS heuristic starts with an initial feasible solution, then selects one of the three removal heuristics, to remove q cargoes from the current solution, and thereafter, selects one of the two insertion heuristics, to reinsert the removed cargoes into the solution. The three removal heuristics are the Shaw removal, the random removal and the worst removal. The two insertion heuristics are the basic greedy heuristic and the regret-k heuristic. Note that the regret-k heuristic is actually a class of k insertion heuristics. We give an elaborate description of the ALNS heuristic, which can be used to solve the basic industrial and tramp ship routing and scheduling problem, in Section 5.2.

Unfortunately, this version of the ALNS heuristic does not provide us with a solution we want, as during the entire route of a vessel its sailing speed is kept fixed and split loads are not allowed. To reduce total operational cost, we want to allow each ship to sail at different speeds, as long as it stays within its range of minimum and maximum speed. The lower the speed of a ship, the lower the fuel consumption of that ship. Therefore, we want to include speed optimization in the ALNS heuristic, which can be done by including an extra algorithm. This algorithm is named the recursive smoothing algorithm (RSA), which solves the speed optimization problem (SOP). Both the algorithm and the formulation of the SOP are provided by Norstad et al. (2011). Besides this algorithm, we also have to include a heuristic which allows one cargo to be split among, thus in other words to be transported by, several vessels. This is the 1-split or merge heuristic proposed by Korsvik et al. (2011). We will describe the extension of the ALNS heuristic in Section 5.3.

5.1 Mixed-Integer Programming Models

In this section we provide the mixed-integer programming models of both versions of the problem. We start in Section 5.1.1 with the mathematical formulation of the basic industrial and tramp ship routing and scheduling problems. Thereafter we continue in Section 5.1.2 with providing the mathematical formulation of the extended problem. As the extended problem will turn out to have non-linear constraints and a non-linear cost function, we also discuss a way to solve this issue. We suggest discretization of the speed such that only integer valued speeds are allowed. This process is explained in Section 5.1.2.1 in detail.

5.1.1 Industrial and Tramp Ship Routing and Scheduling Problems

We strongly rely on Hemmati et al. (2014) and Korsvik et al. (2011) for the mathematical formulation of the basic ITSRSP. Unfortunately, Hemmati et al. (2014) did not include the port cost and the service times at ports, so we slightly reformulate their formulation in line with the formulation of Korsvik et al. (2011). We use the following analogous notation, which is also done in both of these papers.

We let N be the set of cargoes, which we index by i. For each cargo i, there exist a loading port i and a corresponding discharging port $i+n$, with n denoting the total number of cargoes in N. We denote the set of loading ports with $N^P = \{1, \ldots, n\}$, and the set of discharging ports by $N^D = \{n+1, \ldots, 2n\}$. Note that it can be that loading port k and unloading port u ($u \neq k+n$), present the same physical port. We further denote the set of vessels with V, indexed by v. We let N_v be the set of ports which vessel v can visit. In the set N_v, the origin port $o(v)$ and the artificial destination port $d(v)$ for vessel v are included. The artificial destination port $d(v)$ corresponds to the last visited port in the route for vessel v. For each vessel V, we let A_v be the set of arcs it can traverse, thus $A_v = N_v \times N_v$. Lastly, we let $N_v^P = N^P \cap N_v$ and $N_v^D = N^D \cap N_v$ denote the loading and discharging ports which can be visited by vessel v, respectively.

For sailing from port i to port j, we denote the sailing cost by C_{ijv} for vessel v. If a vessel loads or unloads at port i, a port cost has to be paid, which we denote by P_{iv} for vessel v. Further, if cargo i cannot be transported by one of the vessels of our fleet, a cost C_i^S is incurred. In industrial shipping, the cost represents the use of a spot charter to transport the cargo. In tramp shipping, the cost corresponds to the loss in profit from not transporting the optional spot cargo or to the cost of using a spot charter to transport the mandatory cargo. The quantity of cargo i is denoted with q_i, and the capacity of vessel v with Q_v. We let t_{ijv} be the travel time at design speed from port i to port j for vessel v. Each port has a time window $[e_i, l_i]$, with e_i and l_i the earliest and latest start times for loading or unloading at port i. The service time at port i for vessel v is denoted with a_{iv}.

The variables x_{ijv} are binary variables, x_{ijv} equals 1 if vessel v sails from port i to port j and 0 otherwise. The variables y_i are also binary variables, y_i equals 1 if cargo i is transported by a spot charter, and 0 otherwise. Further we let the variable w_{iv} denote the total load on board of vessel v right after the service is completed at port i. Finally, we let t_{iv} denote the start time of the service at port i. The mixed-integer programming model of the industrial and tramp ship routing and scheduling problem (ITSRSP) becomes as follows.

$$\min \sum_{v \in V} \sum_{(i,j) \in A_v} (C_{ijv} + P_{jv}) x_{ijv} + \sum_{i \in N^P} C_i^S y_i, \tag{5.1}$$

subject to

$$\sum_{v \in V} \sum_{j \in N_v} x_{ijv} + y_i = 1, \qquad \forall\ i \in N^P, \tag{5.2}$$

$$\sum_{j \in N_v^P \cup \{d(v)\}} x_{o(v)jv} = 1, \qquad \forall\ v \in V, \tag{5.3}$$

$$\sum_{i \in N_v} x_{ijv} - \sum_{i \in N_v} x_{jiv} = 0, \qquad \forall\ v \in V,\ j \in N_v \backslash \{o(v), d(v)\}, \tag{5.4}$$

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

$$w_{o(v)v} = 0, \qquad \forall\ v \in V, \tag{5.6}$$

$$w_{iv} + q_j x_{ijv} - w_{jv} \leq Q_v (1 - x_{ijv}), \qquad \forall\ v \in V,\ (i,j) \in A_v \mid j \in N_v^P, \tag{5.7}$$

$$w_{iv} - q_j x_{i(n+j)v} - w_{(n+j)v} \leq Q_v (1 - x_{i(n+j)v}), \qquad \forall\ v \in V,\ (i, n+j) \in A_v \mid j \in N_v^P, \tag{5.8}$$

$$w_{iv} \leq Q_v, \qquad \forall\ v \in V,\ i \in N_v, \tag{5.9}$$

$$0 \leq w_{iv} \leq Q_v \sum_{j \in N_v} x_{ijv}, \qquad \forall\ v \in V,\ i \in N_v, \tag{5.10}$$

$$t_{iv} + (a_{iv} + t_{ijv}) x_{ijv} - t_{jv} \leq (l_i + a_{iv} + t_{ijv})(1 - x_{ijv}), \qquad \forall\ v \in V,\ (i,j) \in A_v, \tag{5.11}$$

$$t_{iv} + (a_{iv} + t_{i(n+i)v}) x_{i(n+i)v} - t_{(n+i)v} \leq 0, \qquad \forall\ v \in V,\ i \in N_v^P, \tag{5.12}$$

$$\sum_{j \in N_v} x_{ijv} - \sum_{j \in N_v} x_{j(n+i)v} = 0, \qquad \forall\ v \in V,\ i \in N_v^P, \tag{5.13}$$

$$e_i \leq t_{iv} \leq l_i, \qquad \forall\ v \in V,\ i \in N_v, \tag{5.14}$$

$$y_i \in \{0, 1\}, \qquad \forall\ i \in N^P, \tag{5.15}$$

$$x_{ijv} \in \{0, 1\}, \qquad \forall\ v \in V,\ (i,j) \in A_v. \tag{5.16}$$

The objective function (5.1) minimizes the total cost from operating the fleet plus the cost for using spot charters. Constraints (5.2) make sure that each cargo is transported by either a vessel from the fleet or by a spot charter. Constraints (5.3) - (5.5) describe the flow on the sailing route used by vessel v. The initial load on board of vessel v, which we assume to be zero, is given by (5.6). Constraints (5.7) and (5.8) keep track of the load on board at the loading port and the discharging port, respectively. Further, constraints (5.9) and (5.10) ensure that the load on board does not exceed the capacity of the vessel at loading and discharging ports, respectively. Constraints (5.11) make sure that the start time of service at the discharging port is later than or at the same time as the start time of service at the loading port, plus the travel time from the loading port to the discharging port and the service time at the loading port. Moreover, the loading port has to be visited before the discharging port, which is taken care of by constraints (5.12). Constraints (5.13) ensure further that if a vessel picks up cargo at the loading port, it also delivers the cargo at the corresponding discharging port. Constraints (5.14) make certain that the arrival time at the ports satisfy the time window constraints and lastly, constraints (5.15) and (5.16) set binary restrictions on both the y_i and x_{ijv} variables.

5.1.2 Incorporating Speed Optimization and Split Loads

In order to extend the ITSRSP to incorporate speed optimization, we allow each ship to sail at different speeds on each sailing leg of their scheduled route. Therefore, we introduce a new variable, namely the variable s_{ijv}. This variable denotes the speed of vessel v from port i to port j. Note that the speed of a vessel must stay between the minimum and maximum speed of that vessel. The fuel costs per hour are given by the function $C_v(s_{ijv})$, which is defined in the interval $[s_v^{min}, s_v^{max}]$, with s_v^{min} the minimum speed and s_v^{max} the maximum speed for vessel v. The formula of the cost function $C_v(s_{ijv})$ is given in Section 5.1.2.1. Further, we let d_{ij} denote the distance from port i to port j.

To extend the ITSRSP to incorporate split loads, we allow one cargo to be split among several vessels as it is transported from origin to destination. Therefore, we also introduce the variable z_{iv}, which denotes the quantity of cargo i transported by vessel v. In the mathematical formulation of the ITSRSPSOSL, some constraints remain the same as in the formulation of the ITSRSP, and for those, only a reference to the constraints stated in Section 5.1.1 is given. The mixed-integer programming model of the industrial and tramp ship routing and scheduling problem with speed optimization and split loads (ITSRSPSOSL) becomes as follows.

$$\min \sum_{v \in V} \sum_{(i,j) \in A_v} \big((d_{ij}/s_{ijv})\ C_v(s_{ijv}) + P_{jv}\big) x_{ijv} + \sum_{i \in N^P} C_i^S y_i, \tag{5.17}$$

subject to

$$\sum_{v \in V} \sum_{j \in N_v} x_{ijv} + y_i \geq 1, \qquad \forall\ i \in N^P, \tag{5.18}$$

(5.3) - (5.6),

$$w_{iv} + z_{jv} - w_{jv} \leq Q_v(1 - x_{ijv}), \qquad \forall\ v \in V,\ (i,j) \in A_v \mid j \in N_v^P, \tag{5.19}$$

$$w_{iv} - z_{jv} - w_{(n+j)v} \leq Q_v(1 - x_{i(n+j)v}), \qquad \forall\ v \in V,\ (i,n+j) \in A_v \mid j \in N_v^P, \tag{5.20}$$

$$z_{iv} \leq w_{iv} \leq Q_v, \qquad \forall\ v \in V,\ i \in N_v^P, \tag{5.21}$$

$$0 \leq w_{(n+i)v} \leq Q_v \sum_{j \in N_v} x_{(n+i)jv} - z_{iv}, \qquad \forall\ v \in V,\ i \in N_v^P, \tag{5.22}$$

$$\sum_{v \in V} z_{iv} = q_i(1 - y_i), \qquad \forall\ i\ \in\ N^P, \tag{5.23}$$

$$t_{iv} + a_{iv}x_{ijv} + (d_{ij}/s_{ijv}) - t_{jv} \leq (l_i + a_{iv})(1 - x_{ijv}), \qquad \forall\ v \in V,\ (i,j) \in A_v, \tag{5.24}$$

$$t_{iv} + a_{iv}x_{i(n+i)v} + (d_{ij}/s_{ijv}) - t_{(n+i)v} \leq 0, \qquad \forall\ v \in V,\ i \in N_v^P, \tag{5.25}$$

$$s_v^{min} \leq s_{ijv} \leq s_v^{max}, \qquad \forall\ v \in V,\ (i,j) \in A_v, \tag{5.26}$$

(5.13) - (5.16),

$$z_{iv} \geq 0, \qquad \forall\ v \in V,\ i\ \in N_v^P. \tag{5.27}$$

The objective function (5.17) and constraints (5.18) - (5.22), (5.24) and (5.25) have the same purpose as for the basic model described in Section 5.1.1. Constraints (5.23) ensure for each cargo that the sum of the split quantities is equal to the cargo quantity if the cargo is not transported by a spot charter, and zero otherwise. For the speed variables, constraints (5.26) are lower and upper bounds for each vessel. Lastly, constraints (5.27) make sure that the quantity of cargo i transported by ship v is nonnegative.

5.1.2.1 Discretization of Speed

In the ITSRSPSOSL, the fuel cost ($C_v(s)$) are given in dollars per hour (USD/h) for each vessel v, as a function of the sailing speed s. As done by Brouer et al. (2013), we can express the fuel cost as a non-linear function, given by

$$C_v(s_{ijv}) = \left(\frac{s_{ijv}}{s_v^*}\right)^3 f_v^* \, p^F, \qquad s_v^{min} \leq s_{ijv} \leq s_v^{max}, \qquad \forall \, v \in V, \; (i,j) \in A_v.$$

In this formula, s_{ijv} is the speed in knots (nmi/h) from port i to port j for vessel v, and its defined within the feasible speed range of the vessel. So, the speed s_{ijv} of vessel v, should be between the minimum speed s_v^{min} and maximum speed s_v^{max} of that vessel. The price of the fuel per metric ton (USD/MT) is denoted with p^F, which is in our case equal to 590 USD per MT. Further, s_v^* and f_v^*, the latter in metric tonnes per hour (MT/h), denote the design speed of vessel v in knots and the fuel consumption of vessel v at design speed, respectively.

As this cost function is non-linear, we cannot use this function in our model as we are only able to solve linear models. Therefore, we discretize speed for each vessel within the interval of the minimum and maximum speed of that vessel. For our problem, we only allow each vessel to sail at an integer valued speed in knots. However, note that also smaller intervals could be chosen for the discretization. So in our case, for example, if a vessel can sail between a minimum speed of 10 knots and a maximum speed of 14 knots, we allow that vessel to sail from port i to port j with a speed of 10 knots, 11 knots, 12 knots, 13 knots or 14 knots. Note that a vessel is therefore still allowed to sail at different speeds during the entire planning period, but only allowed to sail at a fixed speed from port to port. In this manner, we are able to compute the costs and travel times of each sailing leg at these different speeds.

In order to incorporate the discretization of speed into the model, we let FS_{vr} be the rth allowed fixed speed of vessel v. To illustrate this, we again take the previous example in which the minimum speed of the vessel is 10 knots and the maximum speed of the vessel is 14 knots. Then, the 1st allowed fixed speed is 10 knots, the 2nd allowed fixed speed is 11 knots, ..., and the 5th allowed fixed speed is 14 knots. We let r range between 1 and the number of possible integer valued speeds within the interval of minimum and maximum speed, and we denote this set with R_v for each vessel v. Using this information, we can compute the rth fixed travel time FT_{ijvr} in hours (h) between port i and j for vessel v by

$$FT_{ijvr} = \frac{d_{ij}}{FS_{vr}}, \qquad \forall \, v \in V, \; (i,j) \in A_v, \; r \in R_v.$$

In this formula, d_{ij} is the distance between port i and j in nautical mile (nmi). As each vessel can only sail one speed at the time on each sailing leg, we introduce the variable ϕ_{ijvr}, which equals 1 if the rth allowed speed is chosen for vessel v for sailing from port i to port j and 0 otherwise. Thus, it must hold that

$$\sum_{r \in R_v} \phi_{ijvr} = x_{ijv}, \qquad \forall \, v \in V, \; (i,j) \in A_v. \tag{5.28}$$

To keep the ITSRSPSOSL formulation the same, we compute the used speed s_{ijv} of vessel v from port i to port j by

$$s_{ijv} = \sum_{r \in R_v} FS_{vr} \phi_{ijvr}, \qquad \forall \, v \in V, \; (i,j) \in A_v. \tag{5.29}$$

Further, we compute the used travel time $T_{ijv} = d_{ij}/s_{ijv}$ by

$$T_{ijv} = \sum_{r \in R_v} FT_{ijvr} \phi_{ijvr}, \qquad \forall \, v \in V, \; (i,j) \in A_v. \tag{5.30}$$

Constraints (5.28) - (5.30) are added to the extended ITSRSPSOSL model described in Section 5.1.2, such that the model becomes linear and solvable with commercial optimization software.

5.2 Adaptive Large Neighborhood Search Heuristic

To be able to solve larger instances, Ropke and Pisinger (2006) proposed an adaptive large neighborhood search (ALNS) heuristic for the pickup and delivery problem with time windows. This heuristic is also used by Hemmati et al. (2014) for the industrial and tramp ship routing and scheduling problem (ITSRSP). Ropke and Pisinger (2006) described the ALNS heuristic as a heuristic that uses several large neighborhoods in an adaptive manner and chooses between them through using statistics gathered during the search. It thereby extends the large neighborhood search (LNS) heuristic, as the latter only incorporates one removal and one insertion heuristic to obtain a solution. Different instances may need different heuristics to obtain a 'better' solution, and therefore it would be more convenient to choose from different heuristics within the process of obtaining a solution. The ALNS heuristic does have a longer computation time compared to the LNS as it implements more heuristics, however it generally does provide better results. The algorithm for the ALNS heuristic is given by Algorithm 1.

Algorithm 1 Adaptive Large Neighborhood Search (ALNS)

```
generate initial solution s
s_best ← s
while stop-criterion not met do
    s' ← s
    select removal and insertion heuristics based on search parameters
    select the number of cargoes to remove and reinsert, q
    remove q cargoes from s'
    reinsert removed cargoes into s'
    if (f(s') < f(s_best)) then
        s_best ← s'
    end if
    if accept (s', s) then
        s ← s'
    end if
    update search parameters
end while
return s_best
```

For the generated initial solution, all cargoes are transported by a spot charter, to ensure feasibility. The cost for the initial solution is thus only the cost for using spot charters to transport the cargoes. The three removal heuristics used in the ALNS heuristic are the Shaw removal heuristic, the random removal heuristic and the worst removal heuristic. All three heuristics remove q cargoes from the current solution. We discuss the removal heuristics in detail in Section 5.2.1. The q removed cargoes will later be reinserted in a new solution by an insertion heuristic. The two insertion heuristics are the basic greedy heuristic and the regret-k heuristic, which we discuss in Section 5.2.2. We note already that the regret-k heuristic is actually a class of k insertion heuristics. A removal and an insertion heuristic are independently selected after weighting the heuristics by an adaptive weight adjustment process. We select a heuristic with a certain probability, by using a roulette wheel selection principle. The selection of the removal and the insertion heuristics is described in Section 5.2.3. Also, an acceptance criterion and a stopping criterion are implemented in the heuristic, which we discuss in Section 5.2.4. Lastly, we apply noise to the objective function such that the heuristic will hopefully not get trapped in a local optimum. We discuss this process in Section 5.2.5.

5.2.1 Removal Heuristics

In this section, we describe the three removal heuristics which are used in the ALNS heuristic. All three heuristics provide a solution in which q cargoes are removed from the current solution. In each iteration of the ALNS heuristic, a new q will be randomly chosen in the interval [1, min(100, $\lceil \xi n \rceil$)]. In this interval, n is the total number of cargoes, and ξ a constant parameter, which we set to 0.4. Note that all of the used parameter values, mentioned here and later on in other sections, are based on the values used by Hemmati et al. (2014) and/or Ropke and Pisinger (2006). However, we let 1 be the minimum value of the interval, while they have both chosen 4 as the minimum value. We cannot use 4 as the minimum value, due to the fact that our smallest instance only consist of 7 cargoes. In that case, q should be chosen from the interval [4, 3], which is not possible.

For each removal heuristic, the current solution is given as input, as well as the number of cargoes q that need to be removed. For the Shaw and worst removal heuristics, also a parameter p is given as input, which will determine the degree of randomization in the heuristics, as stated in Ropke and Pisinger (2006). For the Shaw removal heuristic, we let $p = 6$ and for the worst removal heuristic, we let $p = 3$.

5.2.1.1 Shaw Removal

The main idea in the Shaw removal heuristic is to remove somewhat similar cargoes from the current solution. Ropke and Pisinger (2006) argued that this is convenient because similar cargoes can be shuffled around more easily and thereby find new and (hopefully) better solutions. The similarity of cargoes i and j is measured by the relatedness, which we denote with $R(i, j)$. The relatedness measure is given by

$$R(i,j) = \phi \left(d_{A(i),A(j)} + d_{B(i),B(j)}\right) + \chi \left(|T_{A(i)} - T_{A(j)}| + |T_{B(i)} - T_{B(j)}|\right) + \psi |q_i - q_j|.$$

In this formula, $A(i)$ and $B(i)$ are respectively the loading and the discharging port for cargo i. Further, d_{ij} denotes the distance between ports i and j. We denote the time at which port i is visited with T_i and the quantity of cargo i with q_i.

As can be seen from the formula, the relatedness depends on the distance between the origin and the destination ports, the arrival times at ports and the capacity of the cargoes. The parameters ϕ, χ and ψ are used to weight each of these terms, respectively. We let $\phi = 9$ and $\chi = 3$, as been done by Ropke and Pisinger (2006). Further we let $\psi = 8$, to put more weight on the relatedness in quantities of the cargoes. The relatedness $R(i, j)$ of cargoes i and j is defined in the interval $[0, 2(\phi + \chi) + \psi]$, through normalization of the distances d_{ij}, the arrival times T_i and the quantities q_i. The higher the value of relatedness measure $R(i, j)$, the less cargo i and j are related.

The pseudo code for the Shaw removal heuristic is given in Algorithm 2. At first, a random cargo is chosen and then the similarity with other cargoes is checked by the relatedness measure. The heuristic will remove the initial random chosen cargo and the cargoes that are similar to this random cargo. This process will be continued until all q cargoes are removed from the current solution.

Algorithm 2 *Shaw_ Removal*($s \in$ solutions , $q \in \mathbb{N},\ p \in \mathbb{R}_+$)

cargo : r = a randomly selected cargo from S
set of cargoes : $D = \{r\}$
while $|D| < q$ **do**
  r = a randomly selected cargo from D
  Array : L = an array containing all cargoes from s not in D
  sort L such that $i < j \Rightarrow R(r,\ L[i]) < R(r,\ L[j])$
  choose a random number y from the interval [0,1)
  $D = D \cup \{L[y^p|L|]\}$
end while
remove the cargoes in D from s

5.2.1.2 Random Removal

Out of the three removal heuristics implemented by the ALNS heuristic, the random removal heuristic is the simplest. The heuristic selects q cargoes at random and then removes them from the current solution. The random removal heuristic is basically a special case of the Shaw heuristic, as stated in Ropke and Pisinger (2006), and can be implemented by taking $p = 1$ in the Shaw heuristic. However, we implement the two heuristics separately as the random removal heuristic does have a faster running time if we do not implement it by taking $p = 1$ in the Shaw heuristic.

5.2.1.3 Worst Removal

The last heuristic is the worst removal heuristic. This heuristic removes cargoes with high cost positions, such that it can later be reinserted for lower cost. Therefore, we determine the cost of the cargo in the current solution s by $cost(i, s) = f(s) - f_{-i}(s)$, where $f_{-i}(s)$ is the cost of the solution without cargo i, as given by Ropke and Pisinger (2006). The pseudo code for the worst removal heuristic is provided in Algorithm 3.

Algorithm 3 *Worst_ Removal*($s \in$ solutions , $q \in \mathbb{N}, p \in \mathbb{R}_+$)

while $q > 0$ **do**
  Array : L = All planned cargoes i, sorted by descending $cost(i, s)$
  choose a random number y in the interval [0,1)
  cargo : $r = \{L[y^p|L|]\}$
  remove r from solution s
  $q = q - 1$
end while

5.2.2 Insertion Heuristics

In this section, we describe the two insertion heuristics which are used in the ALNS heuristic. The insertion heuristics reinsert the q cargoes, which were removed by one of the removal heuristics, back into the solution. We first discuss the basic greedy heuristic and thereafter the class of regret-k heuristics.

5.2.2.1 Basic Greedy Heuristic

The basic greedy heuristic is an iterative insertion heuristic. In each iteration of a maximum specified number of iterations, the heuristic inserts one cargo and computes the change in cost of inserting that cargo into a route. If a cargo cannot be inserted in a route, the cost change is set on infinity. The heuristic will insert the cargo at the best position, by taking the position for which the cost change is minimal. This process will be repeated until all cargoes that can be inserted, have been inserted.

5.2.2.2 Regret-k Heuristic

The class of regret-k heuristics improve the basic greedy heuristic, as it looks ahead by selecting the cargo to reinsert into the current solution. In each iteration, the heuristic inserts the cargo at the position for which the regret value is maximized. The regret value is, as stated in Ropke and Pisinger (2006) and Hemmati et al. (2014), the sum of differences in cost of inserting the cargo at its best position, at its second best position, at its third best position, ..., at its k-th best position. When the regret value is maximized for one of the cargoes, this cargo is inserted at its minimum cost position. The basic greedy heuristic can actually be seen as a regret-1 heuristic.

5.2.3 Selection of Heuristics

In order to select both a removal and an insertion heuristic, we weight each heuristic by w_i, with $i \in \{1, \ldots, H\}$. We let H be the total number of heuristics to choose from. In our case, $H = 3$ for the removal heuristics and $H = k$ for the insertion heuristics. We then use a roulette wheel selection principle, to select heuristic j with probability

$$\frac{w_j}{\sum_{i=1}^{H} w_i}.$$

The weights can be given manually, however this can be very time consuming. Therefore, we incorporate an adaptive weight adjusting algorithm, as also been done by Ropke and Pisinger (2006). We strongly rely on their implementation.

In the adaptive weight adjusting algorithm, we measure the performance of a heuristic, by giving it a score π_i. The higher the score, the higher the performance of a heuristic. We divide the search process of the ALNS heuristic into segments, each consisting of 100 iterations. At the beginning of a new segment, we set all scores to zero. The score can be increased, in three different situations, by a given parameter value for that situation. The score of the heuristics that provide a new best solution will increase the most, with a factor of $\sigma_1 = 33$. If the heuristics result in a new solution, and the cost of the new solution are better than the cost of current solution, an increase in the score will be given, with a factor of $\sigma_2 = 9$. Finally, if the heuristics provide a solution which is now accepted but has not been accepted before, and the cost are worse compared to the current solution, the score will be increased with a factor of $\sigma_3 = 13$.

After each segment of 100 iterations, we update the weights w_{ij} of heuristic i in segment j by using the scores π_i. For this, we first normalize the scores by the total number of times we used the heuristic, which we denote by θ_i. The weights of the heuristics in the first segment are all equal, and for the other segments they can be computed by

$$w_{i(j+1)} = w_{ij}(1 - r) + r\frac{\pi_i}{\theta_i}.$$

In this formula, we take $r = 0.1$ as the reaction factor. If the reaction factor is high, the new weights of the heuristics for the next segment will more depend on those of the last segment. If the reaction factor is low, the new weights will stay close to their initial weights.

5.2.4 Acceptance and Stopping Criteria

We implement the same acceptance and stopping criteria as has been done by Ropke and Pisinger (2006). They, and also Hemmati et al. (2014), used the acceptance criterion of simulated annealing. In this criterion, solutions are accepted which are better than the current solutions, but also ones which are worse than the current solution. The latter is accepted with probability $e^{-|f(s')-f(s)|/T}$, in which $f(s')$ and $f(s)$ are respectively the cost of the new and of the current solution, and T is the temperature.

The temperature T is always larger than zero, starts with an initial temperature T_0 and decreases with a cooling rate $c = 0.99975$, through $T = T \cdot c$. The initial temperature T_0 depends on the problem instance and is therefore determined in the same manner as described by Ropke and Pisinger (2006). Namely, first the cost of the initial solution is computed, and then we set T_0 such that a solution which is w percent worse than this solution is accepted with a probability of 0.5. We choose to set $w = 0.05$ as again been done by Ropke and Pisinger (2006).

Finally, as stopping criterion we set the maximum number of iterations of the ALNS heuristic on 25000.

5.2.5 Applying Noise to the Objective Function

Despite that the insertion heuristics are somewhat myopic, Hemmati et al. (2014) do not state that they incorporate any further extensions to deal with this. Fortunately, Ropke and Pisinger (2006) have provided a solution for the fact that the heuristics can get trapped in a local optimum. We strongly rely on their implementation to deal with this problem. In the same manner that we use an adaptive weight adjustment process for the insertion heuristics, we also determine in each iteration whether or not to apply noise to the objective function. By applying noise, in some iterations, to the objective function, we allow for more random solutions. This may seem unnecessary due to the simulated annealing process, however as stated in Ropke and Pisinger (2006), applying noise is very important because the insertion heuristics are not sampled randomly.

To apply noise to the objective function, we first calculate the normal cost C of the insertion of the cargo. We then take a random number $noise$ in the interval [-N_{max}, N_{max}], which we use to compute the modified cost C'. In this notation, N_{max} denotes the maximum noise that can be added to the normal cost C. The modified cost is computed by taking the minimum value of the total cost of not transporting any cargoes by the fleet but by using spot charters, which we denote with C^S and the total of the normal cost C plus the noise factor. So, the modified cost is computed by $C' = \min\{C^S,\ C + noise\}$. Further, we compute N_{max} by $\eta \cdot \max_{i,j \in A_v}\{d_{ij}\}$, such that it depends on the problem size. In our implementation, we take $\eta = 0.025$, as been done by Ropke and Pisinger (2006).

5.3 Incorporating Speed Optimization and Split Loads

In this section, we discuss the solution methods which are used to solve the industrial and tramp ship routing and scheduling problems with speed optimization and split loads. We start in Section 5.3.1 with the explanation of the speed optimization problem. We then propose a recursive smoothing algorithm to optimize speed on a fixed route of a vessel, such that we can extend the basic industrial and tramp ship routing and scheduling problem to include speed optimization. Thereafter in Section 5.3.2, we describe a heuristic which can be used to allow for split loads. This heuristic is called the 1-split or merge heuristic.

5.3.1 Speed Optimization

The ALNS heuristic described in Section 5.2 does not incorporate speed optimization. Fortunately, in the ITSRSPSOSL formulation given in Section 5.1.2, we can recognize a subproblem. The subproblem arises when we assume a fixed route for each vessel. Given the route, the port cost will remain the same even when we change the speed of the vessel on the route. Thus, given the route, the total operational cost will only change if the fuel cost changes by changing the speed. Norstad et al. (2011) also pointed out this subproblem, which they call the Speed Optimization Problem (SOP). We also use this name for the subproblem throughout the rest of our paper.

As stated in Norstad et al. (2011), the objective of the SOP is to minimize the total fuel cost, given the sequence of ports and their corresponding time windows, for each vessel. This can be done by determining the optimal speed for each leg in the given route, such that the fuel cost is minimized. For this, we index each port in the route by $i = 0, \ldots, m$. As the SOP can be solved for each vessel separately, we no longer use the index v in the parameters and variables. However, besides that, we use the same notation as done before, by formulating the SOP. The mathematical formulation of the speed optimization problem (SOP) becomes as follows.

$$\min \sum_{i=0}^{m-1} (d_{i,i+1}/s_{i,i+1})\ C(s_{i,i+1}), \tag{5.31}$$

subject to

$$t_{i+1} - t_i - a_i - d_{i,i+1}/s_{i,i+1} \geq 0, \qquad i = 0, \ldots, m-1, \tag{5.32}$$

$$e_i \leq t_i \leq l_i, \qquad i = 1, \ldots, m, \tag{5.33}$$

$$s^{min} \leq s_{i,i+1} \leq s^{max}, \qquad i = 0, \ldots, m-1. \tag{5.34}$$

The goal is to minimize the value of the objective function (5.31), which is the sum of the fuel cost for a given route of a vessel. Constraints (5.32) make sure that each port is visited in the right order, and that the service at the next port, does not start before the vessel is able to arrive at that port after it completed service at the previous port. The time window constraints are given by constraints (5.33). Note that the first port in the route does not have a specified end time at which the vessel must leave the port, but only an earliest time t_0 at which the vessel becomes available. Finally, constraints (5.34) ensure that the speed of the vessel stays within the feasible speed range of that vessel.

For the SOP two solution methods are proposed by Norstad et al. (2011). In the first method, the arrival time within the time window of each port is discretized. Thereafter, the problem is solved as a shortest path problem. The second method is the recursive smoothing algorithm (RSA), in which the average speed between each port on the route is adjusted until a feasible and optimal solution is found. In our research, we only use the second method proposed by Norstad et al. (2011), as they showed that this method outperforms the first method.

The recursive smoothing algorithm (RSA) is based on the fact that sailing a distance at the lowest possible speed for a vessel, results in the lowest fuel cost for that vessel. This is because the fuel consumption function is convex in the range of feasible speeds. It is therefore also more convenient for a vessel to sail at a constant speed during its entire route, if time windows at each port do not restrict this possibility. For example, disregarding the time windows, we prefer that a ship sails with the same speed from port A to port C, instead of at reduced speed from port A to B and at increased speed from port B to C, even though the arrival time at port C remains the same. The RSA starts with incorporating this example, by determining a fixed speed for the entire route of a vessel. The algorithm then recursively adjusts the speed for each leg of the route until the arrival times at each port, all fall within the time windows for each port. If all arrival times are within the time windows, the solution is optimal.

The full algorithm of the RSA is given in Algorithm 4. In this algorithm, we let A and B present the start and end ports in the route, respectively. Further, t_i denotes the start time of service at port i, a_i the service time at port i and $d_{i(i+1)}$ the distance form port i to port $i+1$. We further let TS denote the total service time on the route. The optimal speed for a vessel, from port i to port j is given by $s^*_{i(i+1)}$, and while the speed is not optimal, we let $s_{i(i+1)}$ denote the speed. Lastly, we denote the time window for port i with $[e_i, l_i]$, with e_i and l_i the earliest and latest arrival times at port i.

Algorithm 4 *Recursive_Smoothing_Algorithm(A,B)*

$\delta \leftarrow 0$
$p \leftarrow 0$
$TS \leftarrow \sum_{i=A}^{B-1} a_i$
$s^*_{AB} \leftarrow \sum_{i=A}^{B-1} d_{i(i+1)}/(t_B - TS - t_A)$
for $i \leftarrow A$ to B **do**
  $i \leftarrow i+1$
  $s_{(i-1)i} \leftarrow s^*_{AB}$
  $t_i \leftarrow t_{i-1} + a_{i-1} + d_{(i-1)i}/s_{(i-1)i}$
  if $t_i - l_i > |\delta|$ **then**
    $\delta \leftarrow t_i - l_i$
    $p \leftarrow i$
  end if
  if $e_i - t_i > |\delta|$ **then**
    $\delta \leftarrow t_i - e_i$
    $p \leftarrow i$
  end if
end for
if $\delta > 0$ **then**
  $t_p \leftarrow l_p$
  Recursive_Smoothing_Algorithm(A,p)
  Recursive_Smoothing_Algorithm(p,B)
end if
if $\delta < 0$ **then**
  $t_p \leftarrow e_p$
  Recursive_Smoothing_Algorithm(A,p)
  Recursive_Smoothing_Algorithm(p,B)
end if

As the ALNS heuristic described in Section 5.2 does not consider variable speed, each new constructed route for a vessel in the solution must be optimized using the RSA, each time the cost of the new constructed route has to be computed. In this manner, both insertion heuristics and the worst removal heuristic will consider the possible cost improvements after optimizing the speed on the route of the vessel. Note that the speed of the vessel must be set on the maximum speed during the rest of the search to check for feasibility of the route, as been done by Norstad et al. (2011), such that the time window constraints at the ports can always be satisfied.

As we only allow integer valued speed in the mixed-integer programming models, we round the optimal speed provided by the RSA up to the nearest integer. In this manner, the time window constraints at the ports are still satisfied. Furthermore, if the optimal speed is lower than the minimum speed of a vessel, we set the optimal speed equal to the minimum speed. By doing this, we are able to compare the results provided by implementing the mixed-integer programming models and the extended version of the ALNS heuristic more accurately.

5.3.2 Split Loads

Besides that the ALNS heuristic described in Section 5.2 does not incorporate speed optimization, it neither does include split loads. In the previous section, we described a way to optimize speed within the ALNS heuristic. In this section, we extend the ALNS heuristic further by allowing for split loads. Among others, Korsvik et al. (2011) discussed a heuristic for this problem and we use this within our extended ALNS heuristic. One of the heuristics that they proposed is a 1-split or merge heuristic, which consist of four steps.

In the first step of the 1-split or merge heuristic, the cargo is removed from all routes it is assigned to. In the next step, all possible insertion points in each vessel route are computed, while satisfying both the capacity and the time window constraints. All these points are sorted by their insertion cost. Thereafter, each combination of routes for which the total spare capacity is equal or more than the quantity of the removed cargo is examined, by inserting the cargo in the cheapest manner on the route. Finally, the combination of insertion points with minimum cost is chosen as solution.

Korsvik et al. (2011) also proposed a 2-split or merge heuristic, as this provides more possibilities for the insertion of the cargo. However, as our ALNS heuristic does already incorporate many heuristics, we only use the 1-split or merge heuristic such that the computation time for the entire ALNS heuristic is not too long.

We implement the 1-split or merge heuristic as one of the insertion heuristics to choose from. The selection of the 1-split or merge heuristic is thus also done by the adaptive weight adjustment process and the roulette wheel principle. Note that the other insertion heuristics, the basic greedy heuristic and the regret-k heuristics, are still only used to reinsert a 'full' cargo into one of the routes. As we implement the heuristic as one of the insertion heuristics, the first step of the 1-split or merge heuristic, in which the cargo is removed from all routes it is assigned to, does not have to be done anymore. This is then actually already been done by one of the removal heuristics.

Chapter 6

Computational Results

In this chapter, we discuss the results obtained by solving the instances in four different ways. The instances were described in Section 4. We first solve the basic industrial and tramp ship routing and scheduling problem (ITSRSP), using both the ALNS heuristic described in Section 5.2 and the model given in Section 5.1.1. Thereafter, we solve the industrial and tramp ship routing and scheduling problem with speed optimization (ITSRSPSO), using both the extended version of the ALNS as described in Section 5.3.1 as well as the model given in Section 5.1.2, without the split load implementation. Then, we use both the extended version of the ALNS as described in Section 5.3.2 and the model given in Section 5.1.2 without the speed optimization implementation, to solve the industrial and tramp ship routing and scheduling problem with split loads (ITSRSPSL). Finally, we solve the full extended industrial and tramp ship routing and scheduling problem with speed optimization and split loads (ITSRSPSOSL), with both the extended version of the ALNS as described in Section 5.3 as well as with the model provided in Section 5.1.2. Solving the instances in these four ways allows for better comparison of the possible cost improvements by extending the basic formulation of the problem.

All mixed-integer programming models are solved using the IBM ILOG CPLEX Optimization Studio Version 12.8, and all heuristics are solved using MATLAB R2017b. By comparing the optimal solutions of the small instances and the lower bounds of the larger instances provided by CPLEX, we get an indication of the performance of the (extended) ALNS heuristic. For each instance, the (extended) ALNS heuristic is executed 10 times, including 25000 iterations in each run. The search for a solution in CPLEX is stopped after one hour, as been done by Hemmati et al. (2014). Both CPLEX and MATLAB were executed on a quad-core, using a 3.4 GHz CPU with 8GB RAM and a 64-bit operating system.

The remainder of this chapter is organized as follows. In Section 6.1, we present the best obtained results of each of the 18 instances for each solution method. We continue in Section 6.2 with discussing the performance of the solution methods. We first discuss the performance of CPLEX and the (extended) ALNS heuristic individually, and thereafter we compare both solution methods. Finally in Section 6.3, we analyze whether or not cost improvements can be made by extending the ITSRSP to more real-life applications, as that is the goal of our thesis.

6.1 Best Obtained Results of Solution Methods

In this section, we discuss the individually best obtained results of the four different models for each instance. These four models are the ITSRSP, the ITSRSPSO, the ITSRSPSL and the ITSRSPSOSL. For each of these models, we provide the results obtained by CPLEX and by the (extended) ALNS heuristic for each of the 18 instances. We only report the best obtained results out of the ten runs of the ALNS heuristic. The known optimal values are indicated by bold face font.

Table 6.1 presents the costs obtained by CPLEX and the ALNS heuristic for the basic industrial and tramp ship routing and scheduling problem (ITSRSP). The results show that CPLEX is only able to solve the model given in Section 5.1.1 for instances which include up to 15 cargoes and 4 vessels, within one hour. The costs provided by the ALNS heuristic are for all instances, evenly as good or better than the costs provided by CPLEX. This means that the same costs are obtained in case of known optimality and otherwise, the cost obtained by CPLEX are higher than the cost obtained by the ALNS. Note that the results provided by CPLEX for the larger instances are very high. This is due to the fact that the only solution found by CPLEX is one in which all, or most of the cargoes are transported using spot charters.

Table 6.1: Results for the industrial and tramp ship routing and scheduling problem (ITSRSP)

		Costs obtained by CPLEX ($)		Costs obtained by ALNS ($)	
Cargoes	Ships	Instance 1	Instance 2	Instance 1	Instance 2
7	3	**1,173,759**	**767,605**	**1,173,759**	**767,605**
10	3	**1,763,098**	**1,530,670**	**1,763,098**	**1,530,670**
15	4	**2,995,352**	3,081,808	**2,995,352**	3,077,774
18	5	3,731,855	3,285,046	3,335,600	3,194,437
22	6	4,329,571	3,893,365	3,272,809	3,465,686
23	13	4,951,880	7,922,644	4,182,256	4,624,710
30	6	5,969,030	8,394,500	3,713,041	4,338,509
35	7	14,048,706	12,162,428	4,981,399	5,108,551
60	13	31,550,695	24,908,704	8,596,102	9,483,378

The Table states the costs for the industrial and tramp ship routing and scheduling problem. On the left side, the Table states the costs obtained by CPLEX after a run time of one hour. On the right side, the Table states the lowest cost obtained by the adaptive large neighborhood search heuristic, out of the 10 runs.

Table 6.2 presents the costs of the industrial and tramp ship routing and scheduling problem with speed optimization (ITSRSPSO). The results show that CPLEX can solve instances with up to around 10 cargoes and 3 vessels to optimality. This is less than for the ITSRSP, which is due to the fact that more variables are included in the model. Note that also due to the same reason, CPLEX is not even able to provide a solution within one hour for the largest instances. For those instances, CPLEX remains in the presolving phase. Fortunately, the ALNS heuristic provides better results, as for most instances the costs are less than those obtained by CPLEX. Note however that the ALNS heuristic does not find any known optimal solution, although it is close to optimal for small instances.

The reason for the latter is that within the ALNS heuristic, we optimize speed over a fixed route using the RSA. As we only allow integer valued speed for the ITSRSPSO in CPLEX, we choose to round the optimal speed obtained by the RSA up to the nearest integer. In that manner, we still satisfy the time window constraint at the ports. However, this can result in higher costs than necessary. For example, if the optimal speed for a vessel is 10.5 knots to sail from A to B to C, then the RSA can determine that the vessel needs to sail at a constant speed of 11 knots from origin to destination. But the optimal speed in CPLEX can be 10 knots from A to B and 11 knots from B to C, which results in lower costs for the route of the vessel.

Finally, note again that the cost obtained by CPLEX for the larger instances are very high as the total cost is the cost of not transporting (most of) the cargoes by the fleet, but by spot charters.

Table 6.2: Results for the industrial and tramp ship routing and scheduling problem with speed optimization (ITSRSPSO)

		Costs obtained by CPLEX ($)		Costs obtained by ALNS ($)	
Cargoes	Ships	Instance 1	Instance 2	Instance 1	Instance 2
7	3	**933,711**	**628,764**	937,823	637,652
10	3	1,539,276	**1,212,183**	1,458,575	1,214,751
15	4	2,108,224	3,215,687	2,250,003	2,096,951
18	5	8,956,417	3,925,740	2,507,575	2,252,206
22	6	12,615,008	6,938,145	2,491,085	2,488,621
23	13	21,239,332	23,061,567	2,977,247	3,236,480
30	6	14,375,843	14,622,333	3,056,477	3,271,521
35	7	19,686,051	23,022,724	3,629,618	4,129,226
60	13	-	-	6,008,301	6,787,264

The Table states the costs for the industrial and tramp ship routing and scheduling problem with speed optimization. On the left side, the Table states the costs obtained by CPLEX after a run time of one hour. On the right side, the Table states the lowest cost obtained by the extended adaptive large neighborhood search heuristic, out of the 10 runs. The extended adaptive large neighborhood search heuristic incorporates the recursive smoothing algorithm to optimize speed.

Table 6.3 presents the costs of the industrial and tramp ship routing and scheduling problem with split loads (ITSRSPSL). For this problem, CPLEX is only able to provide optimal results for the smallest instances. Again, the high costs obtained by CPLEX are the cost of not transporting (most of) the cargoes by the fleet, but by using spot charters. The results obtained by the ALNS are, besides for the smallest instances, all better than the results provided by CPLEX. This means that the costs are lower than the costs provided by CPLEX. The ALNS heuristic does not provide any known optimal solutions, besides the solution of the second instance with the least number of cargoes and vessels. The reason for this will be discussed in the next section, as this is mainly due to the performance of the heuristic(s).

Table 6.3: Results for the industrial and tramp ship routing and scheduling problem with split loads (ITSRSPSL)

		Costs obtained by CPLEX ($)		Costs obtained by ALNS ($)	
Cargoes	Ships	Instance 1	Instance 2	Instance 1	Instance 2
7	3	**1,068,490**	**767,605**	1,073,400	**767,605**
10	3	1,592,748	1,530,670	1,382,423	1,480,551
15	4	3,200,059	4,557,588	2,645,743	2,476,585
18	5	7,597,482	3,720,851	3,058,738	3,220,828
22	6	7,040,378	6,763,137	3,088,453	3,622,013
23	13	13,183,677	14,576,222	4,344,651	4,933,177
30	6	12,727,332	11,237,790	4,009,258	4,633,912
35	7	17,258,154	21,075,254	5,102,405	5,416,075
60	13	32,998,027	32,649,168	9,187,462	9,780,594

The Table states the costs for the industrial and tramp ship routing and scheduling problem with split loads. On the left side, the Table states the costs obtained by CPLEX after a run time of one hour. On the right side, the Table states the lowest cost obtained by the extended adaptive large neighborhood search heuristic, out of the 10 runs. The extended adaptive large neighborhood search heuristic incorporates the 1-split or merge heuristic to make it possible to split loads among several vessels.

Finally, Table 6.4 presents the costs obtained by CPLEX and the ALNS heuristic, for the industrial and tramp ship routing and scheduling problem with speed optimization and split loads (ITSRSPSOSL). For this problem, neither CPLEX nor the ALNS heuristic provide any known optimal results. This indicates that the extended version of the ITSRSP is very hard to solve. For almost all results, the ALNS heuristic provides lower costs than CPLEX. Note that only for the second instance with 7 cargoes and 3 vessels, CPLEX is able to find a better solution. We believe that this is again due to the rounding of the optimal speed up to the nearest integer valued speed within the RSA, just as it was for the solutions of the ITSRSPSO. Note further that also just as it was for the largest instances of the ITSRSPSO, CPLEX is again not able to find a solution within one hour. This time, CPLEX stopped the presolving phase after around 12 minutes for both the largest instances due to an out of memory error.

Table 6.4: Results for the industrial and tramp ship routing and scheduling problem with speed optimization and split loads (ITSRSPSOSL)

		Costs obtained by CPLEX ($)		Costs obtained by ALNS ($)	
Cargoes	Ships	Instance 1	Instance 2	Instance 1	Instance 2
7	3	981,991	635,127	937,820	637,652
10	3	1,709,814	1,605,122	1,130,773	1,192,632
15	4	10,873,189	5,630,471	2,244,790	2,114,839
18	5	12,976,519	11,787,366	2,531,683	2,329,378
22	6	12,615,008	12,133,281	2,715,149	2,832,641
23	13	21,239,332	23,061,567	2,979,297	3,285,446
30	6	14,375,843	14,622,333	3,443,012	3,430,171
35	7	19,686,051	23,022,724	4,141,079	4,202,465
60	13	-	-	7,087,373	7,998,943

The Table states the costs for the industrial and tramp ship routing and scheduling problem with speed optimization and split loads. On the left side, the Table states the costs obtained by CPLEX after a run time of one hour. On the right side, the Table states the lowest cost obtained by the extended adaptive large neighborhood search heuristic, out of the 10 runs. The extended adaptive large neighborhood search heuristic incorporates both the RSA and the 1-split or merge heuristic to optimize speed and to make it possible to split loads among several vessels.

6.2 Performance of Solution Methods

In this section, we discuss the performance of the solution methods. First, the performance of CPLEX and the (extended) ALNS heuristic are discussed individually. Thereafter we compare both solution methods with each other. Note that for all performance gaps, the average is taken over the two instances of the specified number of cargoes and ships.

Table 6.5 presents the average optimality gap and the average gap to the best known solution, of the results provided by CPLEX. The average optimality gap is the average gap of the two instances, between the best solution found in CPLEX and the lower bound. As expected, the larger the size of the problem, the harder to solve to optimality. This can be seen from the increase in the average optimality gap as the size of the problem instances increase. We further see that by extending the problem to either speed optimization, split loads or both, it also becomes harder to solve. This can be seen from the increase in the average gap to the lower bound, and from the increase in the average gap to the best known solution, as the size of the instance increases.

If we compare the average optimality gaps of each problem, we see that the ITSRSP is the easiest to solve, thereafter the ITSRSPSO, then the ITSRSPSL and finally the ITSRSPSOSL. The latter is so difficult to solve for CPLEX that the smallest average gap to the lower bound is 35.00%. While this gap is not reached up to around 22 cargoes and 6 vessels for the ITSRSP. Hemmati et al. (2014) concluded for their implementation of the ITSRSP that optimal solutions could be found for up to around 18 cargoes and 5 vessels. Our implementation is slightly worse, as it is only able to solve instances up to around 15 cargoes and 4 vessels to optimality. However, this can also be due to the fact that we used (slightly) different instances. Note that for the largest instances, the average optimality gap and the gap to the best known solution for the ITSRSPSO and the ITSRSPSOSL is not stated. This is due to the fact that no solution was found within one hour because CPLEX remained in the presolving phase or an out of memory error occurred, for both the ITSRSPSO and the ITSRSPSOSL, respectively.

Table 6.5: Average performance of CPLEX

		Average Optimality Gap (%)				Average Gap to Best Known (%)			
Cargoes	Ships	ITSRSP	+SO	+SL	+SOSL	ITSRSP	+SO	+SL	+SOSL
7	3	0.00	0.00	0.00	35.00	0.00	0.00	0.00	2.25
10	3	0.00	11.74	17.61	58.55	0.00	2.62	8.24	29.78
15	4	11.67	55.71	52.11	89.87	0.07	17.39	31.49	70.90
18	5	33.21	81.00	62.79	92.99	6.69	57.32	36.59	80.36
22	6	43.78	84.12	75.77	89.76	17.70	72.19	51.29	77.57
23	13	60.02	95.27	86.35	95.35	28.58	85.97	66.60	85.86
30	6	62.85	87.74	82.12	88.48	43.06	78.18	63.63	76.30
35	7	74.31	91.60	87.07	92.10	61.27	81.81	72.37	80.36
60	13	83.20	-	88.50	-	67.28	-	70.56	-

The Table presents the average performance of CPLEX over the two instances of each specified number of cargoes and ships. On the left side, the average optimality gap in percentages is stated, which is the average gap from the cost obtained after one hour and the corresponding lower bound. On the right side, the average gap to the best known solution is given, also in percentages.

For the ALNS heuristic, many best known solutions are found for the larger instances. However, if we look at the average of the minimum gap to the best known solution for the smallest instance, the ALNS heuristic only finds solutions without a gap to the best solution for the ITSRSP. This means that we either deal with the problem described in Section 6.1 regarding the ceiling of the optimal speed to get an integer valued speed, or that we may got trapped in a local minimum. The reason for the latter is discussed by Ropke and Pisinger (2006).

Ropke and Pisinger (2006) argued that for different variants of the problem, the parameters used in the ALNS heuristic have to be tuned. This means that parameters should be found which work most efficiently and effectively in finding the optimal solution. We did not perform any parameter tuning, which may have resulted in worse solutions than necessary, as the ALNS heuristic gets trapped in a local minimum. This can mainly be seen from the average gaps to the best known solutions for the ITSRSPSL and for the ITSRSPSOSL, which are on average much higher than those of the ITSRSP and the ITSRSPSO.

Another reason for the large average gaps to the best known solution of the ITSRSPSL and the ITSRSPSOSL is that we did not incorporate the 2-split or merge heuristic as proposed by Korsvik et al. (2011). They argued that the 2-split or merge heuristic provides more possibilities for insertion the cargo into a route. This should lead to better solutions, and thus more cost improvements. A side effect is that the computation time will increase, which lead to our decision to not include this heuristic, as the ALNS heuristic already does incorporate many heuristics and thus has a long computation time.

Table 6.6: Average performance of the (extended) ALNS heuristic

		Average of Minimum Gap to Best Known (%)				Average Gap to Best Known (%)			
Cargoes	Ships	ITSRSP	+SO	+SL	+SOSL	ITSRSP	+SO	+SL	+SOSL
7	3	0.00	0.92	0.23	0.20	0.08	0.93	0.23	0.20
10	3	0.00	0.11	0.00	0.00	0.00	0.11	5.56	5.95
15	4	0.00	3.15	0.00	0.00	0.06	6.58	6.93	10.29
18	5	0.00	0.00	0.00	0.00	1.68	2.70	7.40	9.36
22	6	0.00	0.00	0.00	0.00	1.63	2.90	11.21	15.03
23	13	0.00	0.00	0.00	0.00	2.68	1.94	2.89	4.69
30	6	0.00	0.00	0.00	0.00	4.58	5.44	9.79	9.47
35	7	0.00	0.00	0.00	0.00	4.12	6.42	11.81	15.49
60	13	0.00	0.00	0.00	0.00	4.29	5.20	9.22	12.01

The Table presents the average performance of the (extended) adaptive large neighborhood search heuristic over the two instances of each specified number of cargoes and ships. On the left side, the average of the minimum gap to the best known solution in percentages is stated, which is the average gap from the lowest cost obtained by the (extended) adaptive large neighborhood search heuristic in 10 runs and the best known cost obtained by either CPLEX or the (extended) adaptive large neighborhood heuristic. On the right side, the average gap in percentages to the best known solution is given.

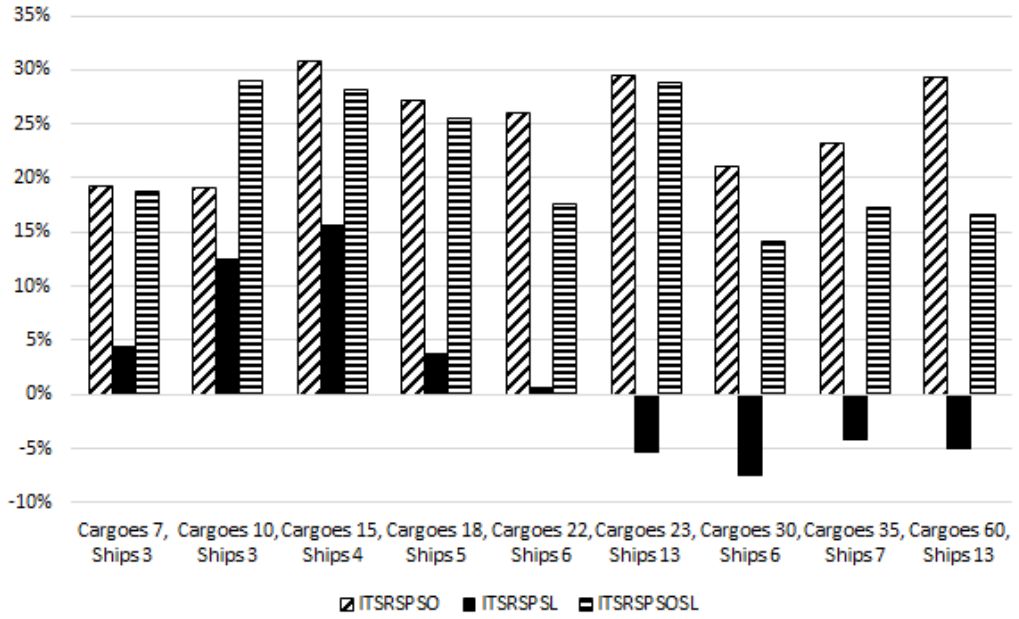
If we compare both solution methods, we conclude that CPLEX should be used for instances which include up to 7 cargoes and 3 vessels, as for most of these small instances better solutions are found. Moreover, with CPLEX we know when a solution is optimal, while with the ALNS heuristic we are not sure if we cannot compare the results with known optimal results. Therefore, it still is convenient to use CPLEX for all instances to compare the results, as we have done. However, the more extensions to the ITSRSP and the larger the size of the problem instance, the more we recommend using the ALNS heuristic to provide solutions. The ALNS heuristic is namely able to solve all instances and it also provides in almost all cases evenly as good or better solutions compared to those obtained by CPLEX.

6.3 Cost Improvements

In this section, we discuss and provide the results of the cost improvements made through extending the ITSRSP to more real-life applications. The goal of our research was to further minimize the operational cost of the fleet by making the extensions. The cost improvements are analyzed for each of the three different variants of the problem. These three variants are the extension of the ITSRSP to speed optimization, to split loads, and to both speed optimization and split loads.

In Section 6.1, all best known results are presented for each solution method and for each instance. For analyzing the cost improvements, we only include the best known result for each instance for each variant of the extended problem. This means that for one instance we may include the solution provided by CPLEX, while for another instance we include the solution provided by the ALNS heuristic. We then take the average percentage of the best known cost improvement of the two instances compared with the best known cost of the ITSRSP, with the same number of cargoes and vessels. These average percentages are shown in Figure 6.1, to allow for good comparison of the cost improvements made through extending the ITSRSP.

Figure 6.1: Percentages of cost improvements



The Figure shows the percentages of the cost improvements made through extending the industrial and tramp ship routing and scheduling problem to speed optimization, split loads and both speed optimization and split loads. The given percentages in the histogram are the average percentages of the two instances of each specified number of cargoes and ships.

We see that on average, most cost improvements are made by extending the ITSRSP to only include speed optimization. This yields on average a cost improvement of 25.02%. However, extending the problem to both speed optimization and split loads also has a huge positive effect on the total costs, as the costs on average reduce with 21.77%. We would expect the cost improvement of the full extension of the ITSRSP, to be higher than the extension to only speed optimization. Unfortunately, the full extended problem is very hard to solve, and due to the fact that we did not perform parameter tuning as discussed in Section 6.2, and because we did not incorporate the 2-split or merge heuristic, we believe that the costs obtained for the full extended problem are further away from the optimal costs than those obtained for the ITSRSPSO. Moreover, we believe that the cost improvements for the extension to split loads and to both speed optimization and split loads, are not significant.

Also, due to time limitations, we could not run more than two instances for each specified number of cargoes and ships. The results of the ITSRSPSL and the ITSRSPSOSL are more instance dependent than those of the ITSRSP and ITSRSPSO. By this we mean that, for each instance, it is possible to sail at a slower speed than the design speed for at least one vessel on a route. However, it is not possible for every instance to split loads among several vessels. This can be seen from the fact that the average cost reduction made by extending the problem to split loads, lies between -7.39% and 15.60%, resulting in an average cost reduction of 1.66%. While for the extension to speed optimization, the average cost reduction over the instances lies between 19.04% and 30.74%. The interval between the minimum and maximum cost improvement is thus smaller for the extension to speed optimization. Therefore, this again indicates that we may not have obtained significant results for testing the extension to split loads.

To show the latter even more, if we only look at the smaller instances up to 18 cargoes and 5 ships, an average cost reduction of 9.06% compared to the costs of the ITSRSP can be obtained for the extension to split loads. Thus, this could mean that large cost improvements can be found (also for the larger instances) by extending the industrial and tramp ship routing and scheduling problem to split loads, if we did not get trapped in a local minimum. This would then also hold for the extension to both speed optimization and split loads, as those cost improvements lie between 14.10% and 28.97% for all instances, but only lie between 19.68% and 28.97% for the smaller instances up to 18 cargoes and 5 ships.

Chapter 7

Concluding Remarks

Industrial and tramp shipping are two of the main shipping industries within seaborne transportation. The objective for those industries is to minimize the operational cost of the fleet by transporting cargoes under time and capacity constraints, in an efficient way. In most literature, it is assumed that within the industrial and tramp ship routing and scheduling problems, the speed of each vessel is fixed (often kept at design speed) and each cargo is only allowed to be transported by one vessel from origin to destination port.

The goal of our thesis was to extend the industrial and tramp ship routing and scheduling problem to more real-life applications, as suggested by Hemmati et al. (2014). We focused on the extensions to speed optimization and to split loads. For the extension to speed optimization, this means that we allow ships to sail at different speeds, as long as they stay within their range of minimum and maximum speed, during the time horizon of the pick-up and deliveries. For split loads, this means that we allow cargoes to be split among several ships such that it can more efficiently be transported. With these extensions, the goal was to further minimize the operational cost of the fleet.

In order to do so, we first reproduced the results of the mixed-integer programming model as proposed by Hemmati et al. (2014), as well as the results of the adaptive large neighborhood heuristic, which was described in detail by Ropke and Pisinger (2006). Therefore, we used the instance generator provided by Hemmati et al. (2014) to create instances based on the data described in Brouer et al. (2013). The results showed that for our implementation of the industrial and tramp ship routing and scheduling problem, CPLEX was able to solve instances with up to around 15 cargoes and 4 vessels, to optimality. The ALNS heuristic was able to find evenly as good, or better results than the results obtained by CPLEX.

To incorporate our extensions into the industrial and tramp ship routing and scheduling problem, we mainly used the insights, the mathematical formulations of the problems and the proposed solution approaches of Norstad et al. (2011) and Korsvik et al. (2011). We first analyzed the extension to only speed optimization. For this extension, CPLEX was able to solve instances with up to around 10 cargoes and 3 vessels to optimality. Unfortunately, the ALNS heuristic in combination with the RSA did not result in any known optimal solutions. However, for most of the instances, better solutions where found by the ALNS heuristic compared to the solutions found by CPLEX. On average, a cost reduction of 25.02% can be obtained by extending the industrial and tramp ship routing and scheduling problem to include speed optimization.

Thereafter, we also analyzed the extension to only split loads. CPLEX was able to find optimal solution for up to around 7 cargoes and 3 vessels. Although we did not perform parameter tuning for the ALNS heuristic, which may have caused worse solutions than necessary, the solutions found by the ALNS heuristic are for most instances better than those found by CPLEX. The extension to incorporate split loads is very instance dependent, which leads to an average cost reduction of only 1.66% compared to the costs of the ITSRSP.

For small instances which consist of a maximum of 18 cargoes and 5 vessels, large cost improvements could be found by the ALNS heuristic for the extension to split loads, namely up to around 15.60% on average. Unfortunately, for the larger instances, the ALNS heuristic got trapped in a local optimum, or was not able to find an optimal solution because we did not incorporate the proposed 2-split or merge heuristic by Korsvik et al. (2011). Incorporating this heuristic would have led to a longer computation time but would provide more possibilities for the insertion of the cargoes.

Finally, we performed the extension of the industrial and tramp ship routing and scheduling problem to both speed optimization and split loads. The results showed again that due to the same reasons as for the extension to split load, the ALNS heuristic got trapped in a local optimum or was not able to find an optimal solution for larger instances due to the fact that we did not incorporate all proposed heuristics. Also, CPLEX was not even able to find an optimal solution for any of the instances, within one hour. This indicates how difficult it is to solve the full extended version of the ITSRSP. On average, a cost reduction of 21.77% could be obtained by extending the ITSRSP to incorporate both speed optimization and split loads.

For further research, we recommend performing parameter tuning on (some of) the parameters used in the adaptive large neighborhood search, as suggested by Ropke and Pisinger (2006). In our research we based the parameters on theirs, however better results can be obtained due to the fact that the parameters are dependent on the kind of variant of the problem. This could be seen from the average gap to the best known solutions, as it seemed that sometimes we get trapped in a local minimum. We also recommend increasing the number of instances, such that more significant results for the cost improvements of the extensions can be given. The latter is mainly needed for the extension to split loads and to both speed optimization and split loads, as we saw that these extensions are very instance dependent. Finally we recommend to also incorporate the 2-split or merge heuristic, which was proposed by Korsvik et al. (2011). Incorporating this heuristic will increase the running time, but it provides more possibilities for insertion the cargo into one of the routes of the vessels. This means that more efficient schedules for pick up and deliveries can be found, which then could imply further cost minimization of operating the fleet.

Bibliography

Andersson, H., Christiansen, M., and Fagerholt, K. (2011). The maritime pickup and delivery problem with time windows and split loads. *INFOR: Information Systems and Operational Research*, 49(2):79–91.

Brønmo, G., Christiansen, M., Fagerholt, K., and Nygreen, B. (2007). A multi-start local search heuristic for ship scheduling—a computational study. *Computers & Operations Research*, 34(3):900 – 917. Logistics of Health Care Management.

Brønmo, G., Nygreen, B., and Lysgaard, J. (2010). Column generation approaches to ship scheduling with flexible cargo sizes. *European Journal of Operational Research*, 200(1):139 – 150.

Brouer, B. D., Alvarez, J. F., Plum, C. E., Pisinger, D., and Sigurd, M. M. (2013). A base integer programming model and benchmark suite for liner-shipping network design. *Transportation Science*, 48(2):281–312.

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

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

Hemmati, A., Hvattum, L. M., Fagerholt, K., and Norstad, I. (2014). Benchmark suite for industrial and tramp ship routing and scheduling problems. *INFOR: Information Systems and Operational Research*, 52(1):28–38.

Korsvik, J. E. and Fagerholt, K. (2010). A tabu search heuristic for ship routing and scheduling with flexible cargo quantities. *J. Heuristics*, 16:117–137.

Korsvik, J. E., Fagerholt, K., and Laporte, G. (2010). A tabu search heuristic for ship routing and scheduling. *Journal of the Operational Research Society*, 61(4):594–603.

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

Lin, D.-Y. and Liu, H.-Y. (2011). Combined ship allocation, routing and freight assignment in tramp shipping. *Transportation Research Part E: Logistics and Transportation Review*, 47(4):414 – 431.

Malliappi, F., Bennell, J. A., and Potts, C. N. (2011). A variable neighborhood search heuristic for tramp ship scheduling. In *International Conference on Computational Logistics*, pages 273–285. Springer.

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

Psaraftis, H. N. and Kontovas, C. A. (2013). Speed models for energy-efficient maritime transportation: A taxonomy and survey. *Transportation Research Part C*, 26:331 – 351.

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

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

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

UNCTAD (2017). *Review of Maritime Transport*. United Nations publication, New York and Geneva. Sales No. E.17.II.D.10.