Erasmus School of Economics

Econometrics & Operations Research

Thesis in Quantitative Logistics

Optimizing cargo flow for tramp and industrial shipping

Name: Kjeld Snoep

Student Number: 424801

Supervisor: Nemanja Milovanovic

Co-reader: Dr. R. Spliet

Abstract

This paper considers a class of deterministic cargo ship routing and scheduling problems with time windows, for several shipping types. To solve these problems, this paper provides an Adaptive Large Neighborhood Search heuristic. This heuristic is evaluated against the CPLEX method that uses a marginal integer program provided in this paper. The heuristic provides high quality solutions on the field of efficiency for all the instances used, with some optimal solution for smaller instances. Furthermore, this paper discusses several methods in cost assignment among customers. Afterwards we compares these methods for several characteristics with each other and give our findings on every methods performances per characteristic.

Contents

1 Introduction

With a current worldwide fleet capacity of about 1.8 billion dead weight tons that carry more than ten billion tons of goods at sea annually (Hoffman, 2017), it can be stated that the maritime transportation is a common transportation way in international trade.

Maritime transportation can be distinguished in three parts: liner, industrial and tramp shipping. In this paper we will only focus on industrial and tramp shipping. Industrial shipping is shipping with cargoes and a fleet that are owned by the operator and it tries to minimize the costs of transporting its own cargoes. Tramp shipping consists of the transportation of some optional and some mandatory cargoes by a fleet, where the main aim is to maximize profits. Bulk goods and oil transport, that contribute for about 60 % of the total transported goods, are almost completely transported via these two ways and therefore these two ways already make up for a significant part of the total maritime transportation.

One of this paper's main focuses lies on cargo routing, which involves the routing of a fleet of ships to service an amount of cargoes that are given as input. In these routing problems, a shipping company has to deal with a set of mandatory cargoes that it has to carry. These cargoes can be a company's own cargoes, but some cargoes that have to be carried because of a contract as well. There are time windows given, within which the loading of the cargoes has to start. There may be time windows for the unloading of the cargoes as well.
The fleet is heterogeneous: each ship has an initial position, time when it becomes available and some ships can not enter some ports due to draft limitations. The aim is to construct a set of routes and schedules such as to minimize costs or maximize profits. Tramp shipping companies may also try to gain revenue by transporting optional cargoes.

Besides cargo routing we also look into the costs allocation of the routes, which involves allocating costs made by the company in a reasonable way for the customers. This problem is among others, researched by Dror (1990) and Potters et al. (1992), who look into cost allocation methods of outcomes from several traveling salesman problems. Frisk et al. (2010) looks into multiple clear methods in costs allocation on a route, while Naber et al. (2015) proposes some methods to implement CO_2 emission allocation for different routes.

It is commonly used to separate deep sea and short sea shipping, where deep sea shipping usually means transatlantic shipping. Furthermore it is common to distinguish between full and mixed load problems as well. In a mixed load problem, ships may carry several cargoes simultaneously while full load problems use the condition that only one load may be carried at once.

There is not much research done on the topic of cargo routing in general, however Dumas et al. (1991) provides a survey from which we can conclude that research in this topic is expanding, with a volume which has doubled in the decade before. A large part of this research in cargo routing is done in the field of industrial and tramp shipping. See for example publications from, Brønmo et al. (2007); Korsvik et al. (2010); Malliappi et al. (2011) that all present a different sort of method to solve this cargo/ship routing problem.
However, there is still not too much information available due to confidentiality issues and a lack of standard benchmark instances that can be used. This is also pointed out by Malliappi et al. (2011). These instances have been developed for several other vehicle routing problems, for say Solomon (1987) for the vehicle routing problem with time windows, and have without a doubt resulted in an increased research interest and several contributions.
Since this survey, among others, Brouer et al. (2013) provided a benchmark suite from real life data from a global liner-shipping company, supplemented by data from other industry and public stakeholders. This benchmark suite is made to reflect the business structure of the global shipping network. This benchmark is presented in relation to business rules and industry standard. They also provide

a mathematical program which is used in this paper.

As it is common to divide the costs made in a route over all the customers in this route, not much research for costs allocation in vehicle routing problem has been done. However, the methods provided for traveling salesman problems give a good basis to work from. We will use these methods to receive some cost allocation results for the instances we solve using a Heuristic.

2 Problem Description

2.1 Cargo Routing Problem with Time Windows

The problem that is dealt with has similarities with the the vehicle routing problem with time windows from Desrosiers et al. (1995). The formulation we used is a bit more detailed. First of all we take the option into consideration that a cargo is transported externally/not transported. Furthermore, we assume that the fleet is heterogeneous, which results in the constraint description used by Hemmati et al. (2014). We will provide you their formulation to gain clarity. A set V consisting all ships is introduced, where ship $v \in V$ has capacity K_v. We let i denote a cargo, there is a node i corresponding to the loading port and a node $i+n$ corresponding to the unloading port. The set of nodes is denoted by N which consists of N^P, the set of loading nodes and N^D, the set of unloading nodes. The set of nodes that can be visited by ship v is N_v, this set includes an origin and a destination node $\big(o(v)$ and $d(v)\big)\big)$. The set of arcs that ship v can cross is A_v. $N_v^P = N^P \bigcap N_v$ are all the loading nodes that can possibly be visited by ship v, and $N_v^D = N^D \bigcap N_v$ are all the unloading nodes that can be visited by ship v. The cost of sailing from i to j using ship v is denoted by C_{ijv} and the related travel time is T_{ijv}. The requested amount at node i is denoted by Q_i Each node has a time window $\big[\underline{T_i}\bar{T_i}\big]$. The time at which ship v's service starts at node i is t_{iv} and l_{iv} is the total load on board of ship v after completing service at node i. The variable x_{ijv} is a binary variable which indicates if ship v moves straight from node i to node j. Binary variable y_i indicates if cargo i is transported externally/not transported. When this is the case, costs will increase with c_I^s. The mathematical problem formulation of Hemmati et al. (2014) is as follows.

$$\min \quad \sum_{v \in V} \sum_{(i,j) \in A_v} C_i jv \cdot x_{ijv} + \sum_{i \in N^p} C_i^S \cdot y_i \tag{1a}$$

subject to

$$\sum_{v \in V} \sum_{(j) \in N_v} x_{ijv} + y_i = 1 \qquad \text{for } i \in N^p \tag{1b}$$

$$\sum_{(j) \in N_v} x_{o(v)jv} = 1 \qquad \text{for } v \in V \tag{1c}$$

$$\sum_{(j) \in N_v} x_{ijv} - \sum_{(j) \in N_v} x_{jiv} = 0 \qquad \text{for } v \in V, i \in N_v \setminus \{o(v), d(v)\} \tag{1d}$$

$$\sum_{(j) \in N_v} x_{jd(v)v} = 1 \qquad \text{for } v \in V \tag{1e}$$

$$l_{iv} + Q_j - l_{jv} \leq K_v \cdot (1 - x_{ijv}) \qquad \text{for } v \in V, j \in N_v^P, (i,j) \in A_v \tag{1f}$$

$$l_{iv} - Q_j - l_{(n+j)v} \leq K_v \cdot (1 - x_{i(n+j)v}) \qquad \text{for } v \in V, j \in N_v^P, (i, n+j) \in A_v \tag{1g}$$

$$0 \leq l_{iv} \leq K_v \qquad \text{for } v \in V, i \in N_v^P \tag{1h}$$

$$t_{iv} + T_{ijv} - t_{jv} \leq (\bar{T}_i + T_{ijv}) \cdot (1 - x_{ijv}) \qquad \text{for } v \in V, (i,j) \in A_v \tag{1i}$$

$$\sum_{(j) \in N_v} x_{ijv} - \sum_{(j) \in N_v} x_{(n+i)jv} = 0 \qquad \text{for } v \in V, i \in N_v^P \tag{1j}$$

$$t_{iv} + T_{i(n+i)v} - t_{(n+i),v} \leq 0 \qquad \text{for } v \in V, i \in N_v^P \tag{1k}$$

$$\underline{T_i} \leq t_{iv} \leq \bar{T}_i \qquad \text{for } v \in V, i \in N_v \tag{1l}$$

$$y_i \in \{0,1\} \qquad \text{for } i \in N^C \tag{1m}$$

$$x_{ijv} \in \{0,1\} \qquad \text{for } v \in V, (i,j) \in A^v \tag{1n}$$

The objective function (1a) should be minimized, this function sums up the costs of the operating fleet plus the cost of external transportation. Constraint (1b) makes sure that all cargoes are picked up. Constraints (1c)-(1e) describe the flow on the route used by ship v. Constraints (1f) and (1g) keep track of the load on board at loading and unloading nodes. Constraint (1h) makes sure that the load of ship v never exceeds its capacity. Equation (1i) makes sure that the time at which service starts is possible, taking travel times into consideration. Constraint (1j) makes sure that if a cargo is loaded, the same ship visits the unloading port. Constraint (1k) makes sure that every cargo that is unloaded could have been loaded before. Time windows are checked by constraint (1l), while constraint (1m) and (1n) are used to make sure that all variables are binary variables. This model can both be used for deep sea and short sea problems.

This model is valid as well for both mixed load and full load cases, though more efficient formulations can be obtained for the full case, see for example the problem description inspired by Christiansen et al. (2007).
This model is basically an extension on the traveling salesman problem, so we can state that this model is NP-hard. As this problem is NP-hard, its runtime will be too long to be convenient for bigger instances. This resulted in us using heuristics in this paper.

In general, there is not much research done on this topic. The multi-vehicle pickup and delivery problem with time windows is at first mentioned by Dumas et al. (1991). The paper does not provide a clear way to solve this problem, and therefore no clear results can be found.

2.2 Cost Allocation

We use cooperative game theory to develop the cost allocation games. We first introduce some notation. Let $N_i = 1 \ldots n$ be the set of customers that are served by vessel i. We have m vessels so

we let i be $\{1 \dots m\}$. A route visiting $S_i \subseteq N_i$ is represented by $\sigma(S_i)$. Let $f(\sigma(S_i))$ be the costs of using a route visiting S_i. After solving the following optimization problem in the first part:

$$\sigma^*(S_i) \in arg\min_i \; arg\min_{\sigma(S_i)} \{ \sum_{1 \dots m} f(\sigma(S_i)) \} \tag{2}$$

To clarify, this equation looks for the optimal routes decomposition in which every cargo is transported, satisfying all the constraints given in Section 2.

The paper of Naber et al. (2015) mainly focuses on the allocation game, so therefore no changes in the visiting order in the route are made, we do exactly the same for the, to be introduced, Lorenz+ Method and EPM+.

Furthermore, it is assumed that $\sum_{j \in N_i} f(j) \geq f(N_i)$, in this way, it is guaranteed that an efficient rational emission exists. However, this is extremely unlikely as there usually are cargoes that are charged for the costs being transported externally. If we look at them individually, all cargoes will be transported using a vessel from within the fleet, as this is cheaper than external transportation. However, it is reasonable to assume that this case holds for every individual route created as the route should be in the right order and ships are usually combined by one vessel to decrease costs in an efficient way. We consider an allocation of costs $x = (x_j)_{j \in N_i}$ to be stable if it is in the core of the game, which is defined next. Using $x(S) = \sum_{N_i} \sum_{j \in N_i} x_j$ the core of a game is defined as

$$core(e) = \{x \in \mathbb{R}^n : x(N_i) = f(\sigma(N_i)); x(S) \leq f(\sigma(S_i)), \forall S_i \subset N_i \text{ for every } i \in \{1 \dots m\}\} \tag{3}$$

As the costs are allocated to customers, we focus mainly on important criteria from the perspective of the customers. First the method is preferred to generate stable allocations. Stable allocations are allocations that lie in the core. For allocations that lie in the core, no subset of customers can gain by withdrawing from the route on the basis of allocated costs. Although We do not assume that customers will withdraw based on their allocated costs, it is preferred that there is no reason to do so either, as it confirms that customers will always gain financially from using the shipping company. Secondly, in order to get an allocation method accepted by a customers and the distribution company, the method should be consistent. That is, if some factor varies, the allocated costs should change accordingly. For example, if the size of a customer's order increases, while other customers' order characteristics do not change, then the costs for the other customers should not increase, and the costs of that one specific customer should not decrease. We also expect an increase when external costs are higher and a cost decrease when the time delivery window expands.

Thirdly, an allocation is preferred to be robust. When a customer receives a similar shipment periodically, it probably does not want to be allocated significantly different costs for every period.
Finally, the runtime for every cost allocation method will be reported. Clearly, companies that frequently have to use an allocation method prefer low computation times.

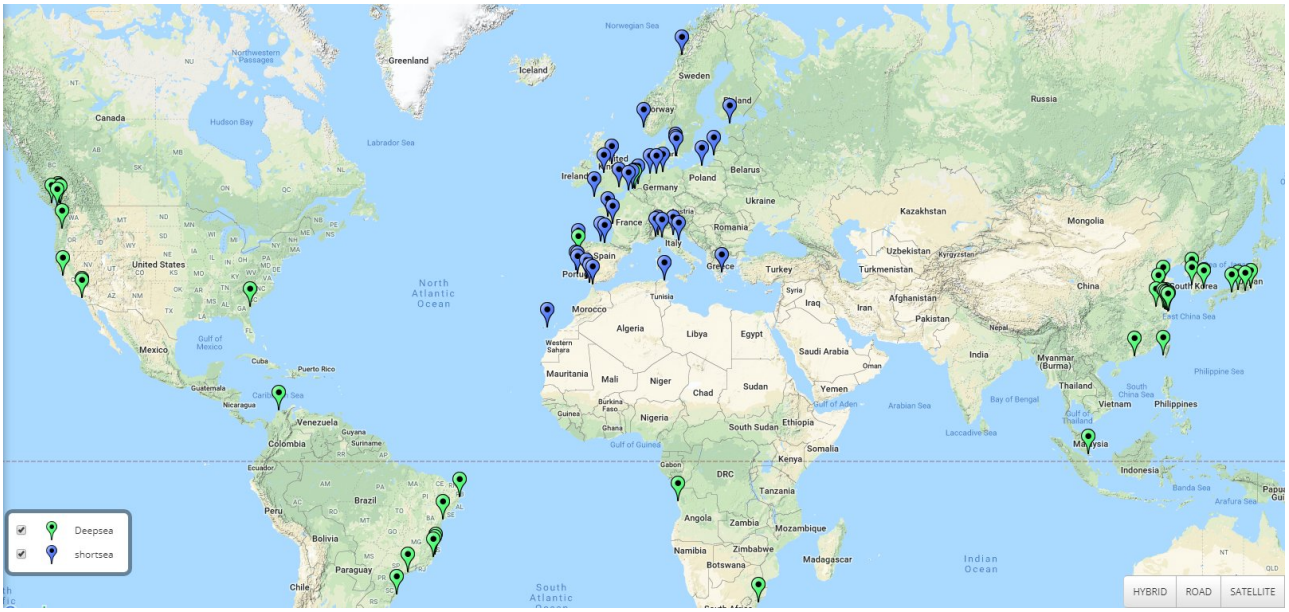
3 Data and Methodology

We used an instance generator[1] that is based on several inputs regarding, cargoes, ships and harbors. Once these inputs have been given, the instance generator works as follows. First, two subsets of ports are generated, with one subset containing harbors where cargoes are often loaded and one

[1] we used instances that are already generated by Hemmati et al. (2014), there instances can be found via `http://home.himolde.no/~hvattum/benchmarks/`, they also provide the best known costs and the best known lower bound for these instances

subset where cargoes are usually unloaded. If the requested instance should have balanced flow of goods, these subsets are equal to the the full set of ports. All these ports are given in figure 1.
Second, a set of cargoes, whose loading and unloading port and size are specified, is generated. After that, the generator makes sure that the selected fleet can service all the cargoes. Thereafter, the time windows and external hiring costs are adjusted according to a selected market situation. In a good market, more cargoes will be available and in a poor market less cargoes will be available. Last, when ships cannot reach any cargo in time for the loading time window, these ships are repositioned such that they can all reach at least one cargo. The generated instances that we have result in a basis for our MIP and ALNS heuristic, which we use to create feasible routes for as little costs as possible.

Figure 1: Figure with all locations used marked



3.1 Adaptive Large Neighborhood Search

An Adaptive Large Neighborhood Search (ALNS) heuristic for a vehicle routing problem was introduced by Ropke and Pisinger (2006). Given below as Algorithm 1, is a general pseudo-code for our ALNS implementation. Algorithm 1 uses an initial solution and then, in every iteration, removes an amount of cargoes from the current solution to reinserts them in a different way to give an other solution. This algorithm is useful as CPLEX results, that use branch and bound to get to an optimal

solution, are known to be inefficient, because of this problem being NP-hard.

Result: solution s which is the best solution found
initialization: generate initial solution s
$s_{best} = s$;
while *stop-criteria is no met* **do**
  $s' = s$
  select removal and insertion heuristic
  select number of cargoes to be removed and reinserted, q
  select q cargoes to remove using the removal heuristic
  reinsert the q removed cargoes into s' using the insertion heuristic
  if $f(s') < f(s_{best})$ **then**
    $s_{best} = s'$;
  end
  if *accept(s',s)* **then**
    $s = s'$;
  end
  update search parameters
end
return S_{best};

Algorithm 1: Adaptive Large Neighborhood Search Heuristic

3.1.1 Initial Start

The ALNS starts with an initial feasible solution. To be sure the solution is feasible, all cargoes are inserted using one of the insertion heuristics for a case where every cargo has to be inserted. In this way we already used an, presumably more efficient way than transporting all cargoes externally.

3.1.2 Removal Heuristics

The removal heuristics are used to remove cargoes from the current solution. In these heuristics, q cargoes are removed from the current solution. In each iteration this q is chosen (randomly). Thus, this number of removed cargoes, q, can vary during the search to provide a different neighborhood size. We decided to choose a q which is an integer within the range $[4, (\epsilon * cargonumber)]$ where we choose a different ϵ for different instances. These epsilons will be given in section Section 4. We will introduce three different removal heuristics introduced by Ropke and Pisinger (2006). The first heuristic is the random removal heuristic which removes q random cargoes in every iteration. The other two methods used are two more complex heuristics. This method is useful as all the cargoes that are removed may be similar and therefore interchangeable.

3.1.2.1 Shaw Removal Heuristic

In the Shaw removal function, at first a random cargo is chosen to be removed. Along with this cargo the $q-1$ cargoes that are most alike are removed as well. This creates a situation where cargoes that can interchange each other easily in the routes are selected. The relatedness is defined with respect to a distance,time, size measure and a measure that takes the ships that can serve the cargoes into account.

$$\begin{aligned} R(i,j) &= \phi(d_{A(i),A(j)} + d_{B(i),B(j)}) + \chi(|T_{A(i)} - T_{A(i)}| + |T_{B(i)} - T_{B(i)}|) \\ &+ \psi(|Q_i - Q_j|) + \omega\left(1 - \frac{|K_i \cap K_j|}{min\{|K_i|, |K_j|\}}\right) \end{aligned} \tag{4}$$

where $d_{i,j}$ is the minimum traveling cost from i to j. The pickup and delivery location of cargo i is denoted by $A(i)$ and $B(i)$, Q_i represents the size of request i. The set of vehicles that are able to transport cargo i is K_i.
In this equation ϕ measures distance, χ measures connectedness of pickup and delivery locations. The term weighted by ψ measures request size and ω measures the relative size of the set of ships that can service both ships, the algorithm used for the shaw Heuristic can be found in Algorithm 2 in the Appendix. In our model we decided to use the same value for every parameter, giving every characteristic the same weight in the similarity check given in eq. (4).

3.1.2.2 Worst Removal Heuristic

The Worst Removal Heuristic has as main idea that the cargoes in the high cost positions are removed. In this heuristic, the costs of cargoes C_i are calculated, after sorting these cargoes in non-increasing order, the first cargo is removed. This procedure is repeated a total amount of q times. The algorithm of this heuristic can be found in Algorithm 3 in the Appendix.

3.1.3 Insertion Heuristics

The paper of Xu et al. (2003) proposes random insertion of the cargos and choosing the best one. Since this method is based on randomness we decided to go for the two insertion heuristics proposed in the paper of Ropke and Pisinger (2006). Both these heuristics reinsert the q removed cargoes, in a parallel way. This means that both heuristics can reinsert cargoes to different ships and therefore create multiple routes.

3.1.3.1 Basic Greedy Heuristic

The basic greedy heuristic is a simple construction heuristic. It reinserts one cargo in each iteration. We introduce $\Delta f - i, k$ which denotes the change in the objective value of reinserting cargo i into route k. If we can not insert request i in route k the we set $\Delta f_{i,k} = C_i^S$ which is the price of external/no delivery. We then introduce c_i as $c_i = \min_{k \in K}\{f_{i,k}\}$ which is the lowest cost of inserting cargo i in a ship. Finally we choose the cargo i that such that.

$$\arg\min_{i \in U} c_i \tag{5}$$

and insert this cargo at its minimum cost position. U is the set of removed cargoes. This process repeats itself till all removed cargoes have been inserted. However, we do not have to repeat the entire process as only one route changes and therefore $\Delta f_{i,k}$ only changes in this route. This can be used to speed up the heuristic.

3.1.3.2 Regret Heuristic

The Regret Heuristic tries to improve the greedy heuristic by already looking at information after inserting a cargo. Let $x_{ik} \in \{1, \ldots, m\}$ be a variable that indicates request k has the k-th lowest insertion cost. that is $\Delta f_{i,k} \leq \Delta f_{i,k'}$ for $k \leq k'$. This method can express c_i from 3.1.3.1 as $c_i = \Delta f_{i,x_{i1}}$. In the regret heuristic we define a regret value c_i^* as $c_i^* = \Delta f_{i,x_{i2}} - \Delta f_{i,x_{i1}}$. so basically, the regret value represents the cost difference between inserting the cargo in the best and the second best position. In every iteration the regret heuristic uses a cargo such that it maximizes c_i^* for $i \in U$. The request is inserted in its minimum cost position. Ties are broken by selecting the insertion with lowest cost. We basically choose the cargo which we will regret not picking first most. The heuristic

can be extended in a way such that it become a Regret-k Heuristic. In the Regret-k Heuristic the construction heuristic in every construction step chooses to insert the cargo i that maximizes:

$$\max_{i \in U} \{ \sum_{j=1}^{k} (f_{i,x_{ij}} - f_{i,x_{i1}}) \} \tag{6}$$

The request is inserted at its minimum cost position. The first explanation of the Regret Heuristic provided in this section is a Regret-2 Heuristic. Compared to a regret-2 Heuristic, Regret Heuristics with a higher value of k discover earlier that the possibilities for inserting a request becomes limited. These heuristics are useful as they take later stages of insertion into consideration as well,

3.1.4 Selection of Heuristics

We have multiple heuristics that can be used in the optimization, besides choosing one heuristic we can always decide to use multiple heuristics in one optimization. We can do this using a roulette selection procedure with a probability assigned to every heuristic. After creating the heuristics we have 9 insertion heuristics and 3 removal heuristics. We assign an initial probability to them which is equally distributed in the case of the insertion heuristics and where the star method has an initial starting probability to be picked of 60% and the other two removal heuristics both have a probability of 20%. We use the Adaptive Weight Adjustment method proposed by Ropke and Pisinger (2006). For updating the scores after every 100 itterations, we first normalize the scores (π_i) by dividing them by the number of times they have been used (θ_i) and afterwards, the new weights will be calculated by using a predefined parameter (r) to balance the previous weights and the new normalized scores as follows. The new weights become:

$$w_{i,s+1} = w_{i,s}(1 - r) + r\frac{\pi_i}{\theta_i} \tag{7}$$

Where $w_{i,s}$ is the weight of i used in segment s. After weighing the heuristics, we have k heuristics with weights wi, $i \in \{1, 2, ..., k\}$ then we select heuristic j with the probability given in equation Equation (8) by using a roulette wheel selection principle. Note that the removal and insertion heuristic are selected independently.This selection procedure is efficient because it creates a bias towards more successful removal and insertion heuristics.

$$p_{j,s} = \frac{w_{j,s}}{\sum_{i=1}^{k} w_{i,s}} \tag{8}$$

3.1.5 Acceptance and Stopping Criteria

We use simulated annealing to decide whether a solution with higher costs is accepted or not. Solutions with lower costs will always be selected. We select a solution with higher costs with a probability $e^{\frac{new-f}{T}}$ with $T > 0$. Where T is the temperature. We decided to set the initial temperature as the biggest costs difference and iteratively update this following Kirkpatrick et al. (1983). We do this to create a more dynamic model and prevent our heuristic from ending up in a local minimum.
A stopping criterion will be set in the amount of runs. Note, the best solution will always be saved in case the final solution is not the best solution found.

3.2 Cost Assignment

After creating the routes we look deeper into the possibility of assigning the total costs made a to the different customers. Inspired by the CO_2 emmision allocation methods of Naber et al. (2015)

we decided to implement the Star method, the Lorenz+ Allocation method and the Equal Profit Method+.

3.2.1 Star Method

The Star Method allocates the costs proportionally to the stand-alone costs of every customer. The allocation amount is equal to:

$$x_i = \frac{e(\{i\})}{\sum\limits_{(j)\in N} e(\{j\})} e(N) \quad \text{for } i \in N \tag{9}$$

Where $e(\{i\}$ represents the costs of only transporting cargo i, in this way the costs of the cargoes will be allocated equally based on relative values of the individual costs.

In practice, instead of using the stand-alone costs in every route used it may as well be morally responsible to look at the minimal stand-alone costs over the vessels or to use the external costs and compare them. In this way, customers do not get duped for not being part of a route, or being part of an inconvenient route.

This method is commonly used for the traveling salesman problem, because it has a low runtime and is easy to understand for every party. However, allocations generated by this method are not necessarily stable. This method could result in outcomes that lie outside of the core, even when the core is non-empty. This is the case for the traveling salesman problem, so it may as well be the case if we do calculate the costs as if the groups of customers in a vessel are separated from the other customers for core calculations.

3.2.2 Lorenz+ Allocation

The Lorenz Allocation was introduced as Leximin by Arin (2007). It is the core allocation with the smallest difference between the largest and smalles cost allocation to any customer within the same route. The Lorenz allocation is the optimal solution to the following LP problem, where f represents the smallest difference in allocated costs:

$$\begin{aligned}
\min \quad & f && && \text{(10a)}\\
\text{s.t.} \quad & x_i - x_j \leq f && \text{for } i,j \in N && \text{(10b)}\\
& x_S \leq e(S) && \text{for } S \subset N && \text{(10c)}\\
& x_N = e(N) && && \text{(10d)}
\end{aligned}$$

This method does not provide a solution when the core is empty as (10c) & (10d) define the definition of the core. When this set of x is non/existent, there is no answer. This makes it either unsuitable for a multiple routes, or the routes would not all be given their individual costs in total. Therefore, we decided to allocate the costs created in every route separately. When the core for individual routes is empty, we use the Star emission in these specific routes, as this method also works for a non empty core, this is why we call this method the Lorenz+ method. This method can be used by companies if distributional equality is preferred. Furthermore, the optimal solution of this LP problem is often not unique.

3.2.3 Equal Profit Method+

The Equal Profit Method (EPM) given by Frisk et al. (2010) provides an allocation similar to the Lorenz+ Allocation. However, instead of minimizing the largest absolute difference between customers, we minimize the largest relative difference between the allocation compared to the minimum stand-alone cost of each customer as calculated mentioned in the Star method. The EPM+ allocation is the optimal solution to the following LP problem, where g represents the largest relative difference:

$$\min \quad g \tag{11a}$$

$$\text{s.t.} \quad \frac{x_i}{e(\{i\})} - \frac{x_j}{e(\{j\})} \leq g \quad \text{for } i, j \in N \tag{11b}$$

$$x_S \leq e(S) \quad \text{for } S \subset N \tag{11c}$$

$$x_N = e(N) \tag{11d}$$

Like the Lorenz+ allocation, the EPM+ does not provide an allocation when the core is empty. We use the Star Method if the core is empty, hence we call it the Equal Profit Method+, or EPM+ allocation for short. Again, this allocation is not unique in general.

4 Results

We solved all our routing initially both using the ALNS heuristic described in section 2 and CPLEX using the problem described in section 2 in AIMMS. However, CPLEX could not give solutions for the biggest instances within an acceptable amount of time. Therefore, we decided to mainly compare the results of the heuristics to the best result possible. We hereby assume that the CPLEX method can not be implemented because the runtime exceeds every reasonable threshold such that it will never be useful to apply in real life. To confirm this we ran both our heuristic and the CPLEX for an hour for certain instances, this gave results that confirm our thoughts about CPLEX. Furthermore we do not report runtimes for all ALNS instances, as it is hard to compare cases with each other as we adjusted some parameters in our model for bigger instances to save some runtime. However, all the results we received took less than 9 hours of runtime and the smallest instances were usually finished running within 60 seconds.

4.1 Method Specifications

In order to avoid the heuristic from taking up to much runtime, we decided to change some parameters as our cases got bigger. This resulted in a slightly worse program. However, the runtime decreases outweigh the performance decreases. Therefore, we decided to update the parameters for different instance sizes. Table 1 shows all the values we used for these different sizes. Furthermore, we decided to choose an r of 0.5 for all instances in our weight adjustment method described in 3.1.4, which we update every 100 iterations.

Table 1: Parameters Used

vessels	1-34	35-50	51-99	100+
epsilon	0.75	0.5	0.5	0.5
iterations	2000	1200	1000	500

Table 2: ALNS Results for Deep Sea Shipping Instances with Mixed Cargo Loads

cargoes	vessels	instance #1	instance #2	instance #3	instance #4	instance #5
7	3	**5233464**	**6053699**	**5889056**	6510656	**7220458**
10	3	7986248	7986248	**9566942**	**8617192**	**8785219**
15	4	**13795202**	**13886935**	12677334	13620168	**10833641**
18	5	**50564519**	**37341527**	**30365579**	32281904	43336197
22	6	49451327	43507540	49214036	55430668	54277296
23	13	41814173	42066086	39543390	36504765	44358844
30	6	25862397	21535658	22227781	28349400	34499834
35	7	81311960	83474363	63059145	84599655	107834236
60	13	97398238	139323183	109932316	134754981	
80	20	122806413		135396230	137241195	116846065
100	30		326144018		180642612	189216231
130	40	416479975		393482847	363288000	500945985

Bold face font indicates the highest lower bound costs known and the underlined numbers indicating that the outcome is within 10% range of the highest lower bound costs known . Blank spaces represent instances that have not been run.

Table 3: ALNS Results for Deep Sea Shipping Instances with Full Cargo Loads

cargoes	vessels	instance #1	instance #2	instance #3	instance #4	instance #5
8	3	9816881	9472892	**4596681**	7109977	6815253
11	4	34891635	25460327	**29627143**	34149233	**28175914**
13	5	12015246	13321117	10948497	14512578	11875425
16	6	51633161	48032194	51971383	43708016	53702018
17	13	17652867	13046211	**12091554**	15847689	13293621
20	6	16582364	17341381	18011412	18257796	19309402
25	7	24887777	25831017	22432920	26667503	27763782
35	13	97393953	102147165	101316800	95788951	115049913
50	20	50815142	41598898	47144565	50786018	52770698
70	30					
90	40					
100	50	223559813	311429727	235234127		309872871

Bold face font indicates the highest lower bound costs known costs and the underlined numbers indicating that the outcome is within 10% range of the highest lower bound costs known. Blank spaces represent instances that have not been run.

4.2 Computational Results

The ALNS was executed on a HP elitedesk 705 G1 SFF with 4GB RAM and a Clock Rate of 3.5 GHz using matlab, for the CPLEX problem, AIMMS was used. ALNS results for all instances are reported in tables 2 to 5. The CPLEX results of AIMMS were not able to give a feasible result for every instance within in hour. Also, for large instances, these solutions usually involve a lot of servicing the cargoes externally, which leads to higher costs. We can see in table 13, where we compared CPLEX and the ALNS after running both for 1 hour for some instances, that for the small instances CPLEX gives good outcomes within a reasonable amount of time. However, for the bigger instances, the ALNS method appears to outperform the CPLEX solver from AIMMS.

We can see in tables 6 and 7 that for the small instances, ALNS provides anwers that lie within a 10% range of the best known answer for most of the time, and this difference gets bigger as we introduce some larger instances. However, for the bigger instances, table 13 shows that the ALNS method appears to outperform the CPLEX solver of AIMMS, especially for the bigger instances.

Table 4: ALNS Results for Deep Sea Shipping Instances with Mixed Cargo Loads

cargoes	vessels	instance #1	instance #2	instance #3	instance #4	instance #5
7	3	**1476444**	**1134176**	**1196466**	1657162	**1160394**
10	3	2204593	2020521	**1986779**	**2125461**	**2162453**
15	4	**1959153**	**2560004**	2600046	2272223	**2230861**
18	5	**2374420**	**2987358**	**2301308**	2400016	2842413
22	6	4345671	3848843	3365795	3423552	3868111
23	13				2282593	2376839
30	6	5581051		4652612	4593968	5134789
35	7	5456592	5242213	5051545	4895518	6631823
60	13			8549032	9274121	10489016
80	20	16870053	18715375	16591819	14874311	
100	30	23263266	19556582		16404034	22575785
130	40	26722207	22793646	24696474		22575785

Bold face font indicates the highest lower bound costs known and the underlined numbers indicating that the outcome is within 10% range of the highest lower bound costs known

Table 5: ALNS Results for Short Sea Shipping Instances with Full Cargo Loads

cargoes	vessels	instance #1	instance #2	instance #3	instance #4	instance #5
8	3	**1391997**	**1246273**	**1698102**	**1777637**	1657668
11	4	1080236	1068722		1204743	1471639
13	5	2503697	2061318	**2378283**	2727367	3079761
16	6	4065271			3775951	3587380
17	13	2376051			2820346	2924255
20	6	2999655	3254475	3211554	3350835	3191797
25	7	3908254	3809208	4299252	4333730	4146765
35	13	3095227	3149647	3183766	4278768	3529406
50	20	7800073	8106774	7515034	11814613	8419378
70	30					
90	40		16205433			
100	50	16791721	23107559	20005940	24089051	23094789

Bold face font indicates the highest lower bound costs known and the underlined numbers indicating that the outcome is within 10% range of the highest lower bound costs known. Blank spaces represent instances that have not been run.

Table 6: Comparison of Instances with Mixed Cargo loads Best Known Solution and the ALNS Answers

		short sea shipping	deep sea shipping
cargoes	vessels	ALNS fraction of best known	ALNS fraction of best known
7	3	1,0639	1,0000
10	3	1,0124	1,0104
15	4	1,0020	1,0611
18	5	1,0021	1,1536
22	6	1,0537	1,2903
23	13	1,0219	1,2223
30	6	1,1068	1,2862
35	7	1,1343	1,3921
60	13	1,1720	1,4259
80	20	1,6024	1,6915
100	30	1,5425	1,4761
130	40	1,4390	1,7412

The fraction indicates the outcome of the solvers solver divided by the best solution known

Table 7: Comparison of Instances with Full Cargo Loads between Best Known Solution and the ALNS Answers

		short sea shipping	deep sea shipping
cargoes	vessels	ALNS fraction of best known	ALNS fraction of best known
8	3	1,0026	1,0131
11	4	1,0418	1,0065
13	5	1,0539	1,1030
16	6	1,0698	1,1185
17	13	1,0194	1,0658
20	6	1,0084	1,0678
25	7	1,0214	1,1854
35	13	1,0543	1,1712
50	20	1,1422	1,1840
70	30	Not a Number	Not a Number
90	40	1,1590	Not a Number
100	50	1,5087	1,3178

The fraction indicates the outcome of the solvers solver divided by the best solution known, 'Not a Number' is used for the instances where we do not have any results to provide.

4.3 Cost Allocation Method Performances

We mention the results stated in the paper of Naber et al. (2015) and give comments on the differences in our case(vehicle routing in stead of traveling salesman problems).

4.3.1 Stability

A non-empty core was used for every instance in the case study of Naber et al. (2015). Because the Lorenz+ and EPM+ guarantee a solution in the core when it is non-empty, their allocations were stable by definition. In contrast, their allocation of the Star method resulted in an allocation in the core in 34.9% of the instances. So from a stability point of view, the Star method performs worst.
In the case of multiple routes, the Lorenz+ and EPM+ method would still have allocation that are stable. However, this is only the case if the routes are seen as separated parts, otherwise an empty core is a possibility that is very likely, and the case for all the instances given on the because of the fact that external transportation is always needed and more expensive.

In our case less than 2% of the instance had a non-empty core for every vessel's route in it, so we may say that the assignments of the Star method, that never satisfied the core constraint is only performing a little bit worse than the other methods. Besides, our definition of the core was only used within a route for the methods, because non of the cases had a non-empty core when all the routes were combined. Based on this we can conclude that none of the methods were very stable.

4.3.2 Consistency

Consistency is evaluated by Naber et al. (2015) based on the distance to the depot, the average distance to other customers and the order size. These factors (possibly) influence the allocated costs. Therefore, they performed an OLS regression to evaluate the consistency of every allocation method, where the allocated costs is the dependent variable. The explanatory variables were the distance to the home depot, the average distance to other customers, the size of the order and a constant. The allocated costs were used as the dependent variable. Furthermore, they added two cross-product variables that included the order size. They did this because these cross-terms were combined in the CO2 emission calculation. It turned out that, except for three cases, the coefficients of the variables were significantly different from zeros at a 5% significance level. However, the size had a negative coefficient in the Lorenz+ method, which is not expected. Furthermore, the coefficients of the cross products were both positive and negative. So these results, may be ignored. However, it can be stated that the Lorenz+ method is very likely to be inconsistent on the size basis. The Star method and EPM+ did have a significant positive effect on the allocated emission, so, we have reason to believe that they are consistent in general. In addition, they state that the R-squared of the Star Method is the highest and therefore they mark the Star Method most consistent.

In the case of multiple routes it may be the case that some routes are inconsistent compared to other routes for every method as costs can be significantly higher for some cargoes due to extern transportation. However, we used a regression of the cost allocation using, weight, average time window length and external costs as variables (tables 8 to 11). If we use a 95% significance interval we can see that, for all three methods, extra weight and higher external costs mean higher costs. However, a bigger size of the time window and therefore more flexibility only resulted in significant negative coefficient once, this is when we use the Star method. Also, the significance of the other two variables was usually higher which makes it safe to say that we assume the Star method to be the most consistent one. This thought gets confirmed by the R-squared values of the methods, as it has the highest value for every instance. Tables 8 to 11 also show that this value is a little bit higher for the EPM+ method, ergo the EPM+'s consistency may be a little bit higher than the Lorenz+ method's consistency, which is also stated by Naber et al. (2015) .

Table 8: Regression on the Allocations of the Deep Sea Instances with Full Cargo Sizes

	Star Method		Lorenz+ Method		EPM+	
	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
Constant	12059.52	0.7213	642442.7	0.0008	719157.8	0.0003
external costs	0.342051	0.0000	0.301664	0.0000	0.299993	0.0000
size time window	51.51994	0.0774	-230.1441	0.1718	-294.7142	0.0881
R-squared	0.973895		0.271742		0.259952	

Table 9: Regression on the Allocations of the Short Sea Instances with Full Cargo Sizes

	Star Method		Lorenz+ Method		EPM+	
	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
Constant	22481.50	0.0117	-122131.3	0.0001	-131966.2	0.0000
external costs	0.371862	0.0000	0.231719	0.0000	0.235717	0.0000
size time window	-53.45197	0.0061	319.0046	0.0000	332.3661	0.0000
R-squared	0.773029		0.188462		0.192463	

Table 10: Regression on the Allocations of the Deep Sea Instances with Mixed Cargo Sizes

	Star Method		Lorenz+ Method		EPM+	
	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
Constant	39837.07	0.2457	31926.36	0.8016	-88511.97	0.5387
external costs	0.294789	0.0000	0.250972	0.0000	0.321458	0.0000
size time window	54.09246	0.0040	71.02026	0.2959	-26.00561	0.7340
weight	1.239981	0.0000	4.790137	0.0000	4.047357	0.0000
R-squared	0.743579		0.180970		0.184744	

Table 11: Regression on the Allocations of the Short Sea Instances with Mixed Cargo Sizes

	Star Method		Lorenz+ Method		EPM+	
	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
Constant	-3079.024	0.3617	-23951.00	0.1344	-37745.74	0.0202
external costs	0.325058	0.0000	0.259937	0.0000	0.271782	0.0000
size time window	-4.443799	0.4310	4.689872	0.8608	16.00059	0.5555
weight	0.686295	0.0000	5.856402	0.0000	5.863635	0.0000
R-squared	0.790633		0.153608		0.159982	

4.3.3 Computation time

We provide the average computation times per instance for all the allocation methods used. The computation time is low for every method, Naber et al. (2015) provide the average computation times for all the allocation methods they used. The computation time is low for all allocation methods, however the Star method is more than 15 times faster than the Lorenz+ and EPM+. This is due to the fact that the star method has a polynomial number of computations, while the LP problems have an exponential number of constraints. Clearly, the size of the instances is still relatively small, so the running times are not too high. In our case we can sea ass well that the STAR method's running time is significantly lower than the running time of the Lorenz+ Method, there is also a notable difference between the running time of the Lorenz+ Method and the EPM+ method. To conclude, based on runtimes we can state that the STAR Method performs best and the EPM+ performs worst.

Table 12: Average Runtime per Instance

allocation method	time (s)
STAR Method	7.15
Lorenz+ Method	13.19
EPM+	23.86

5 Conclusion, Limitations and Future Research

We have considered the cargo ship routing and scheduling problem with time windows, which arises in industrial and tramp shipping. We presented up to 240 benchmark instances that represent realistic planning problems four several segments of this problem. An instance generator to create more cases has been provided by Hemmati et al. (2014). We used an ALNS heuristic to solve the cargo ship routing and scheduling problem with time windows. The benchmark instances have been solved by the ALNS and a MIP solver. The MIP solver is able to find optimal solutions for the smaller instances given. The ALNS solver usually finds solutions that are at least as good as those given by the MIP solver, and that can be used for big instances as well, as runtime is low and the answers are almost always within a 50% range of the best known answer. After these solutions are found we use the STAR, Lorenz+ and EPM+ method to allocate costs between the customers in the every instance. Comparing these methods showed us that on the field of consistency and Run time, the STAR method outperforms both the other methods, while the other methods perform slightly better on the field of stability.

However, the model and program we created have some limitations. The code used to solve this ALNS had some efficiency issues, this may lead to higher runtime and therefore, we were not able to run all the instances provided.
Furthermore we are aware that this ALNS uses deterministic values for the shipment times, which does not always seem reasonable given the real life situation.
Besides this we basically used a two step approach in deciding how to divide the costs over several cargoes in which we first create a route and then assign cost to each cargo afterwards. However, it may be more stable if we could create a method in which the assignment of the costs is not only decided based on the final route of the ALNS, but if it also uses other route compositions, that may lead to a more stable devision of costs between the cargoes that are transported externally and internally.

We hope our work will stimulate researchers to create algorithms for this important planning and devision problem and, even more importantly, make these instances more realistic, such that these algorithms work even better in real-life.

References

Arin, J. (2007), 'Egalitarian distributions in coalitional models', *International Game Theory Review* **9**(01), 47–57.

Brønmo, G., Christiansen, M., Fagerholt, K. and Nygreen, B. (2007), 'A multi-start local search heuristic for ship schedulinga computational study', *Computers & Operations Research* **34**(3), 900–917.

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. (2007), 'Maritime transportation', *Handbooks in operations research and management science* **14**, 189–284.

Desrosiers, J., Dumas, Y., Solomon, M. M. and Soumis, F. (1995), 'Time constrained routing and scheduling', *Handbooks in operations research and management science* **8**, 35–139.

Dror, M. (1990), 'Cost allocation: the traveling salesman, binpacking, and the knapsack', *Applied Mathematics and Computation* **35**(2), 191–207.

Dumas, Y., Desrosiers, J. and Soumis, F. (1991), 'The pickup and delivery problem with time windows', *European journal of operational research* **54**(1), 7–22.

Frisk, M., Göthe-Lundgren, M., Jörnsten, K. and Rönnqvist, M. (2010), 'Cost allocation in collaborative forest transportation', *European Journal of Operational Research* **205**(2), 448–458.

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.

Hoffman (2017), Review of maritime transport, Technical report, UNCTAD, New York and Geneva.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983), 'Optimization by simulated annealing', *science* **220**(4598), 671–680.

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.

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', Springer, pp. 273–285.

Naber, S., de Ree, D., Spliet, R. and van den Heuvel, W. (2015), 'Allocating co2 emission to customers on a distribution route', *Omega* **54**, 191–199.

Potters, J. A., Curiel, I. J. and Tijs, S. H. (1992), 'Traveling salesman games', *Mathematical Programming* **53**(1-3), 199–211.

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.

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations research* **35**(2), 254–265.

Xu, H., Chen, Z.-L., Rajagopal, S. and Arunapuram, S. (2003), 'Solving a practical pickup and delivery problem', *Transportation science* **37**(3), 347–364.

A Appendix

A.1 Removal Heuristics

Algorithm 2: Shaw Removal

function ShawRemoval($s \in \{solutions\}, q \in \mathbb{N}, p \in \mathbb{R}_+$)
  request : r = a randomly selected request from S
  set of requests : $D = \{r\}$
  while $|D| = \{r\}$ **do**
    r = a randomly selected request from D
    Array: L = An array containing all request from s not in D
    Sort L such that $i \leq j \rightarrow R(r, L[i]) < R(r, L[j])$
    choose a random number y from the interval [0,1)
    $D = D \cup \{L[y^p|L]\}$
    end
  remove the requests in D from s
end function

Algorithm 3: Worst Removal

function WorstRemoval($s \in \{solutions\}, q \in \mathbb{N}, p \in \mathbb{R}_+$)
  while $q \geq 0$ **do**
    Array: L = All planned requests i, sorted by descending $cost(i, s)$
    choose a random number y from the interval [0,1)
    request : $r = L[y^p|L]$
    remove r from from solution s; q=q-1;
    end
end function

A.2 Tables

Table 13: Comparison CPLEX and Heuristic

Deepsea/Shortsea	full/mixed cargo sizes	#cargoes	#vessels	Cplex answer	Heuristic Answer	Best Answer Known
Deepsea	full	13	5	11820655(0.5 sec)	11820655	11820655
Deepsea	full	20	6	16529748(256 sec)	16580233	16529748
Deepsea	full	50	20	62257674	43482561	41398100
Deepsea	full	70	30	513166677	177746128	162903901
Deepsea	full	100	50	667681419.00	258344112	224430601
Deepsea	mixed	15	4	12457251(126 sec)	12457251	12457251
Deepsea	mixed	22	6	36459842	34462012	34129809
Deepsea	mixed	35	7	209579470	66607715	65082675
Deepsea	mixed	80	20	360503449.00	97510237	78918099
Deepsea	mixed	130	40	no solution found	285584294	246883618
Shortsea	full	13	5	2043253 (1.19 sec)	2043253	2043253
Shortsea	full	17	13	2806231 (226,95 sec)	2811427	2806231
Shortsea	full	35	13	3215480	3061701	2986667
Shortsea	full	70	30	38328124	12172906	10314521
Shortsea	full	100	50	49702794	15902779	14106741
Shortsea	mixed	15	4	2560004 (74 sec)	2560004	2560004
Shortsea	mixed	22	6	3507448	3307557	3228262
Shortsea	mixed	35	7	18387821	5413762	4942430
Shortsea	mixed	80	20	42930718	12116948	9763401
Shortsea	mixed	130	40	no solution found	21418183	18533293

No solution found indicates that no feasible solution was found after running aimms for an hour.