A Branch-and-Price Approach to the Feeder Network Design Problem

Santini, Alberto; Plum, Christian Edinger Munk; Røpke, Stefan

Published in:
European Journal of Operational Research

Link to article, DOI:
10.1016/j.ejor.2017.06.063

Publication date:
2018

Document Version
Peer reviewed version

A Branch-and-Price Approach to the Feeder Network Design Problem

Alberto Santini, Christian E.M. Plum, Stefan Ropke

PII: S0377-2217(17)30604-5
DOI: 10.1016/j.ejor.2017.06.063
Reference: EOR 14545

To appear in: *European Journal of Operational Research*

Received date: 14 June 2016
Revised date: 24 June 2017
Accepted date: 27 June 2017

Please cite this article as: Alberto Santini, Christian E.M. Plum, Stefan Ropke, A Branch-and-Price Approach to the Feeder Network Design Problem, *European Journal of Operational Research* (2017), doi: 10.1016/j.ejor.2017.06.063

Highlights

- The problem of selecting customers and designing routes on a container shipping feeder network is proposed.
- The model takes into account many real-life characteristics and constraints.
- The model is solved with a branch-and-price algorithm.
- Realistic instances are produced, starting from publicly-available benchmark instances.
- Computational results and a scenario analysis are presented, to provide insights on desirable features of optimal routes.

A Branch-and-Price Approach to the Feeder Network Design Problem

Alberto Santini[a,*], Christian E. M. Plum[c], Stefan Ropke[b]

[a]*Deutsche Post Chair of Optimization of Distribution Networks, RWTH Aachen University, Kackertstraße 7b, 52072 Aachen, Germany*

[b]*DTU Management Science, Produktionstorvet 424, 2800 Kgs. Lyngby, Denmark*

[c]*Mærsk line, Esplanaden 50, 1098 København, Denmark*

Abstract

In this paper we consider the problem of designing a container liner shipping feeder network. The designer has to choose which port to serve during many rotations that start and end at a central hub. Many operational characteristics are considered, such as variable leg-by-leg speeds and cargo transit times. Realistic instances are generated from the LinerLib benchmark suite. The problem is solved with a branch-and-price algorithm, which can solve most instances to optimality within one hour. The results also provide insights on the cost structure and desirable features of optimal routes. These insights were obtained by means of an analysis where scenarios are generated varying internal and external conditions, such as fuel costs and port demands.

Keywords: OR in maritime industry, network design, liner shipping, branch and price, vehicle routing problem

1. Introduction

Container liner shipping is a freight transportation service used to move large quantities of cargo over long distances. As opposed to tramp shipping, that describes individual ships operated to fulfil the transportation requests as they come, and that decide which one is more convenient for them to accept, liner ships operate along fixed routes and according to a published schedule.

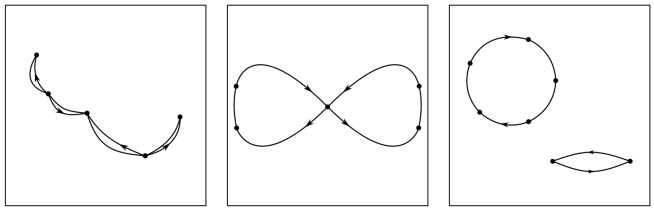
According to Lloyd's [22], 75% of internationally traded goods by volume was transported by sea in 2008 and this figure is set to increase: it was already an estimated 85% during 2013, according to Drewry [17]. Containerised goods account for a relatively small portion of the shipped volume, but their value per volume unit is higher than that of any other kind of goods, and around 52% of maritime commerce by value is shipped in containers.

Furthermore, the environmental impact of moving goods by sea is certainly non-negligible: World Shipping Council [37] estimated that 2.7% of the global greenhouse gas emissions is accounted by international maritime shipping, and a quarter of this figure is due to container shipping.

These data hint at the potential impact of optimisation of liner container vessels operations. Such optimisations can yield enormous results in terms of both business and environmental value. And yet, operational research methods have not frequently focused on liner shipping. This is especially clear when comparing

*Corresponding Author

Email addresses: santini@dpo.rwth-aachen.de (Alberto Santini), chrplum@gmail.com (Christian E. M. Plum), ropke@dtu.dk (Stefan Ropke)



(a) Conveyor Belt (b) Butterfly (c) Cycle and Pendulum

Figure 1: Different types of rotations. Reprinted from Andersen [2] with permission.

the amount of literature relevant to this field versus other fields of logistics and transportation (see, e.g., Christiansen et al. [11]).

In this paper, we consider the Feeder Network Design Problem (FNDP), which arises when planning the routes of liner container vessels in regional feeder networks. Intercontinental container routes are operated by big vessels (up to 18000 TEU, Twenty-foot Equivalent Units) that only call the main hub ports. These hubs are characterised by an extensive infrastructure that makes them suitable to operate with enormous quantities of containers, and to efficiently load and unload extremely big vessels. Networks of smaller vessels load the containers delivered at the hubs and transport them to smaller ports in a particular region. At the same time, they collect containers at the small ports and unload them at the hubs; from there, they will later be shipped to their final destination on the intercontinental routes. In short, liner shipping is organised in *hub-and-spoke* networks. While more than one hub can exist in the same region, in this work we focus on single-hub feeder networks.

Since liner vessels operate according to a fixed schedule, the operator issues a public timetable indicating when each port will be visited in each rotation. It is convenient for shippers and supply chain agents that such a schedule be periodic, so that each port is called on a certain day of the week that does not change. For this reason, the time horizon considered when planning feeder routes is a multiple of one week.

Due to the nature of the feeder network, moreover, each schedule starts and ends at the hub port (even though it can visit the hub more than once). For this reason, we usually refer to the route taken by a vessel as a *rotation*. Rotations are sometimes classified according to their structure: Figure 1, reprinted from Andersen [2], depicts cycles, pendulum routes, butterfly routes, and conveyor belts. In our work, we allow the creation of routes with arbitrary shapes, visiting the hub any number of times within the time horizon. Therefore, all rotation types presented in Figure 1 are allowed to be constructed, as particular cases of our routes.

The network designer is also faced with the decision of selecting which ports to serve. The liner company earns revenue for serving each port. The company might also be requested to pay a penalty when not serving a port, if it had contracts in place with terminal or logistic operators. Another possibility is that the company decides to outsource the service to another company (for example, by buying capacity on a competitor's vessel). As we will show in the following sections, our model is flexible enough to allow for each of these scenarios.

We assume that both the delivery and pickup demands of a port are determined and known in advance. In practice, the data we use comes from forecast of future demand, possibly based on historical observation or contracts in place. We also assume that, for each port, the network designer has two decisions to make: whether or not to pickup demand at the port, and whether or not to deliver demand to the port (these decisions are independent). Once the decision is made to serve, e.g. the pickup demand at the port, the total amount of cargo will be picked up in one visit. In other words, we do not allow split pickup and delivery.

The quantity to maximise is the total revenue from served cargoes, minus the cost of the rotations and the eventual penalties or outsourcing costs. The cost of rotations includes port taxes and calling fees and the

vessel's bunker cost, i.e. the cost of the fuel used by the ship. This latter cost is particularly important for two reasons: first, it makes up a considerable share of an operator expenses, while its price remains very volatile; second, it is greatly impacted by the steaming speed of a vessel, with the relationship between the two being approximately cubic:

$$\text{cost}(s) = \left(\frac{s}{s^*}\right)^3 \cdot \text{cost}(s^*)$$

where s is the steaming speed, $\text{cost}(s)$ is the cost incurred to sail for a unit of time at speed s, s^* is the design speed, and $\text{cost}(s^*)$ is the cost to sail for a unit of time at the design speed. Both s^* and $\text{cost}(s^*)$ are constants which depend on the vessel's characteristics.

The constraints the operator is faced with are (1) the limited capacity and number of vessels available; (2) the fact that ports observe certain closing time windows (e.g. ports that are closed for operations at night); (3) that certain goods might have a maximum time span during which they can travel (e.g. perishable goods) and (4) that ports have a maximum draught and therefore not every vessel can enter every port.

Link to routing problems

The FNDP asks to come up with certain routes for a fleet of vehicles (vessels, in this case) that abide particular constraints. The problem is, therefore, related to the well-known Vehicle Routing Problem (VRP) and many of its variants: it combines elements of the *capacitated* VRP since each vessel has a maximum container capacity, the VRP with *time windows*, the VRP with *heterogeneous fleet* since each vessel can differ from the others, the VRP with *pickups and deliveries* since we have a flow of cargo both from and to the hub, the *multi-period* VRP because the vehicles can get back to the hub multiple times. We refer the reader to the recent book by Toth and Vigo [34] for a comprehensive review of various VRP variants.

In these variants of the VRP, however, the set of customers to be served is given and no selection need be performed. With this respect, the FNDP is then related to the family of Orienteering Problems (OP), and in particular to the multi-vehicle variants which are present in the literature under various names, including Team Orienteering Problem (TOP), VRP with Profit Collection, or VRP with Customer Selection. The TOP has received considerable attention in the recent years (see, for a review, Archetti et al. [4]). While exact algorithms based on branch-and-price exist for the TOP [7] and for its capacitated version [3], to the best of our knowledge no exact algorithm has been developed for cases which include time windows, or a pickup-and-delivery component.

The main contributions of this paper are three-fold. First, we introduce a mathematical model and an extended formulation for the Feeder Network Design Problem, taking into account the principal real-life constraints that container vessel operators have to face. Second, we provide a state-of-the-art algorithm, which is able to solve to optimality (or with very small gaps) realistic instances. Third, we perform a wide variety of scenario analyses, from which we distill general principles an operator can apply at the strategic, tactical, and operational levels.

2. Literature review

In this section we summarise recent literature on two important aspects of maritime optimisation: liner network design, and speed optimisation. For more general reviews of research in maritime optimisation, we refer the reader to Christiansen et al. [11], Christiansen et al. [12] and Christiansen et al. [13].

Liner network design

Liner shipping describes the transportation of goods by sea vessels which follow fixed routes and published timetables. This is opposed to tramp shipping, where ships receive service offers from brokers and make

Criterion	Value	Criterion	Value
Optimisation Criterion	Profit	Shipping Market	Liner
Decision maker	Owner	Explicit fuel price	Yes
Freight rate	Yes	Fuel consumption	Cubic
Leg-by-leg optimal speed	Yes	Speed function of payload	No
Logistical context	Pickup and delivery	Size of fleet	Multiple ships
More ships an option	Yes	Inventory costs	No
Modal split	No	Ports included	Yes

Table 1: Classification of the present paper under the taxonomy proposed by Psaraftis and Kontovas [29].

an "on-line" decision on which offers to take. By feeder network design, we mean a particular case of liner network design, where the ships make up the feeder part of a larger hub-and-spoke network.

An introduction to liner shipping is given in Brouer et al. [8], where the authors also present a benchmark model for LinerLib (see Løfstedt et al. [23]), the main instance library used for liner shipping problems. The benchmark model for the Liner Network Design Problem (LNDP), is solved by means of a heuristic based on Tabu search and column generation. A broad introduction to operational research methods in container liner shipping is given in the survey by Meng et al. [25].

A heuristic, a column generation algorithm and a Benders decomposition algorithm, all coupled with an iterative search routine, are presented by Agarwal and Ergun [1] to solve the combined ship scheduling and cargo routing problem, with weekly frequencies and transhipments. The authors allow multiple visits to the same port (as long as they happen in different days of the week), but do not consider transhipment costs. They test their approach on instances with at most 20 ports and 100 ships.

An exact method is presented by Reinhardt and Pisinger [31] to solve the network design and fleet assignment problem. The authors use a branch-and-cut algorithm to solve instances up to 15 ports, while considering transhipments, a heterogeneous fleet and allowing butterfly routes.

Mulder and Dekker [26] propose a heuristic approach to the combined problem of fleet design, ship scheduling and cargo routing. The authors first cluster ports by proximity and then design a hub-and-spoke hierarchy, in which each cluster is served by a feeder network. They test their approach on the Asia-Europe trade lane, comprising 58 ports.

Plum et al. [28] propose to replace the classic arc-flow formulation for the LNDP with a service-flow based formulation. Similarly to our work, they consider the Baltic and Western African scenarios of the LinerLib and propose a mixed-integer linear formulation to maximise the operator's profit, under a weekly frequency constraint and allowing multiple calls to the same port (thereby allowing an arbitrary number of butterfly ports). The model is solved with a commercial solver.

Another heuristic method, based on column generation, for the network design and cargo flow problem in liner shipping is presented by Wang and Meng [36]. The authors apply a column generation algorithm to a mixed-integer non-linear non-convex formulation, which models weekly services and maximum transit times (which they refer to as *deadlines*). They test their approach on the Asia-Europe trade lane, considering 12 ports.

Plum et al. [27] present arc-flow and path-flow models for the single-vessel liner shipping service design, in which an operator has to optimise the best-paying demand (pickups and deliveries) for a round-trip route operated by a single vessel, taking into account maximum transit times. The authors propose a branch-and-cut algorithm, which is able to solve instances with up to 25 ports.

Speed optimisation

A general review and a taxonomy of existing literature related to speed optimisation for efficient and green maritime transportation is given in Psaraftis and Kontovas [29]. Table 1 shows the classification of our

work under the proposed taxonomy.

Chang and Wang [10] and Cariou [9] study the economical impact and sustainability of slow steaming in liner shipping. In the first work, the authors conclude that speed optimisation is best employed as a dynamic process, which depends on both charter rates and fuel prices. In the second work, the author concludes that slow steaming is a long-term sustainable strategy on main container trade lanes, if the bunker price is at least \$350-\$400.

More details on the relation between sailing speed, incurred costs and emissions is given in Psaraftis and Kontovas [30] and Kontovas [21], in which the authors study the trade-offs involved in speed optimisation in broader shipping scenarios. In particular, Psaraftis and Kontovas [30] consider many real-life factors impacting on the relationship between speed and costs, stressing the importance of considering hotel costs to adjust the purely cubic cost function, the dependency of the cost function on the payload, the impact of external events such as weather conditions, and of the maintenance state of the ship (e.g. hull condition). They also mention the importance of *en route* inventory costs, which clearly tend to increase at lower speeds on long-distance routes. Kontovas [21], on the other hand, focuses on the relationship between fuel consumption and CO_2 emissions.

The work of Wang and Meng [35] studies the leg-by-leg optimisation of sailing speeds, once the routes have already been fixed. The author consider transhipments and container routing, and propose a Mixed-integer non-linear convex model, for which they give an outer approximation scheme. They test their approach on an instance with 46 ports and 11 fixed routes.

Placement of this work

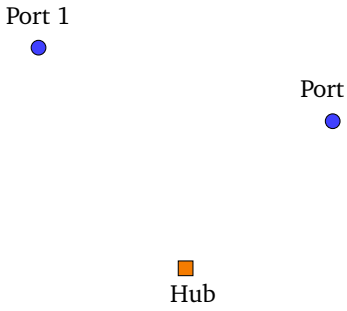
The focus on this work is on feeder networks, and therefore we need not consider transhipments. This allows for in-depth modelling of the network itself, to a degree which has not been done before in the literature. In particular, we combine strategic decisions (port selection, fleet utilisation) with tactical ones (speed optimisation) in a single optimisation framework, which also takes into account multiple heterogeneous vessels and port characteristics. The results presented in Section 6 show that our approach allows to solve instances of similar size to those solved in the literature, while employing a more detailed and richer model.

3. Model

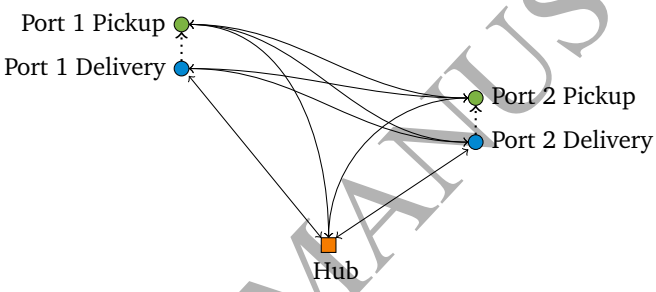
The time in the planning horizon is divided into m time intervals, and it is modelled by the set $T = \{1, \dots, m\}$. The size of this discretisation depends on the size of the region considered. As mentioned in Section 1, it is convenient that the time horizon be a multiple of one week. If, for example, we consider one month as planning horizon, we can then deploy four ships on each rotation spacing them out by one week, so that each port is visited weekly.

Let $n \in \mathbb{N}$ be the number of ports in the region, excluding the hub. We model each port twice, once as a pickup and once as a delivery port. The set of all ports is denoted by $P' = \{0, 1, \dots, n, n+1, \dots, 2n\}$ where 0 represents the hub, $P^- = \{1, \dots, n\}$ is the set of delivery ports, and $P^+ = \{n+1, \dots, 2n\}$ is the set of pickup ports. Ports i and $n+i$ represent the same physical port.

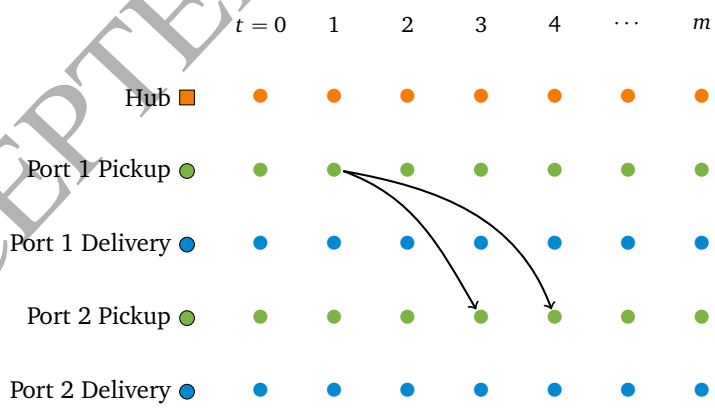
We also define the set $P = P' \backslash \{0\}$ of all ports excluding the hub. Each port $i \in P^-$ has a delivery demand $d_i \geqslant 0$ and each port $j \in P^+$ has a pickup demand $p_j \geqslant 0$. Furthermore, each port has a maximum draught $m_i^{\mathrm{p}} > 0$, meaning that vessels with a draught greater than m_i^{p} cannot enter the port. Each port has a handling time $h_i \geqslant 0$, that is the number of time intervals needed to load (if $i \in P^+$) or unload (if $i \in P^-$) containers at that port. Since the quantity of containers to be moved at each port is known in advance, an accurate estimate of h_i can be given for each port. We assume that when the vessel visits the hub, its



(a) Configuration of the instance, with the ports in their geographical positions.



(b) In this graph, the ports have been duplicated (for pickup and delivery) and arcs represent the possibility to sail from one port to another.



(c) The time-expanded graph for a particular vessel, where each node is copied once for each time instant. Two arcs are depicted, representing sailing between two ports at different speeds.

Figure 2: Modelling of a simple instance, with one hub and two ports.

handling time h_0 can be estimated only based on the vessel type, independently from the quantity of cargo it carries (this quantity, in fact, cannot be estimated in advance).

Each port has a number of closing time windows which can be used not only to model times at which the port is closed, but also for many other purposes. Here we give a couple of examples.

- If i is a pickup port that receives most of the goods it exports (for example, via a freight train) at a certain time $t_0 > 1$, then i can be marked as closed until a time $t_1 \geqslant t_0$ at which we can assume the goods are ready to be loaded onto a ship.
- If we use a time horizon representing 2 weeks, and have a delivery port j which is served weekly by a freight train (at times t_0, t_1), and which imports perishable goods that can stay in the yard at most t' time units before they are loaded on the train, we can mark the port as open only during time windows $[t_0 - t', t_0]$ and $[t_1 - t', t_1]$. A rotation will visit j, for example, during the first time window; the second ship deployed on the same rotation will visit it during the second time window, thereby guaranteeing a weekly visit, and that the goods are not spoiled by staying too long in the yard.

For every pair of ports i, j we have a symmetric distance $\Delta_{i,j} \geqslant 0$. The distances satisfy the triangle inequality, and are zero for elements modelling the same port, i.e. $\Delta_{i,i'} = 0$ for all $i \in P'$, and $\Delta_{i,n+i} = 0$ for all $i \in P^-$.

We deal with a heterogeneous fleet of k vessels, modelled by set $V = \{1, \ldots, k\}$. Each vessel v has a capacity $Q_v > 0$ and a draught $m_v^{\mathrm{v}} > 0$. We consider the draught of a vessel as fixed, even though in reality it can change slightly with the amount of load a vessel is carrying. We refer the reader to, e.g., Glomvik Rakke et al. [19], Battarra et al. [6], Malaguti et al. [24] for recent works on maritime routing under draught limits. Each vessel v has a set of possible sailing speeds $\Omega^v = \{\omega_1^v, \ldots, \omega_{s_v}^v\}$ expressed in nautical miles per discrete time unit (i.e., the speed is expressed in multiple of knots); $s_v \in \mathbb{N}$ is the number of speeds associated with vessel v.

We take into account the following cost components:

- Time charter cost: a fixed daily cost that the operator pays to charter the vessel. We associate to each vessel v the time charter cost Γ_v^{TC} (in dollars per time interval).
- Hotel cost: the cost incurred when keeping a vessel moored, but operational. We denote the hotel cost for vessel v as Γ_v^{H} (in dollars per time interval).
- Bunker cost: the sailing cost, which depends on the cruising speed of the vessel. Given a vessel v and a sailing speed $\omega_k^v \in \Omega^v$, we consider the associated cost Γ_{vk}^{B} (in dollars per time interval).
- Movement costs: the unit costs relative to the handling of the cargo at a port. Given a vessel v and a port i, since we know the quantity of cargo that needs to be handled in i, we can compute the total movement cost and denote it as Γ_{vi}^{M} (in dollars).
- Fixed port call fee: a flat cost to pay when any vessel calls a port i. This is denoted as Γ_i^{PF} (in dollars).
- Variable port call fee: a cost to pay when calling a port, but dependent on the capacity of the vessel. A vessel v calling port i will pay $\Gamma_{vi}^{\mathrm{PV}}$ (in dollars).

Serving requests generates revenue, as the operator earns a certain fee per Forty-Foot Equivalent Unit (FFE) picked up or delivered. The revenue is, for commercial reasons, not uniform and delivering an FFE to a certain port could earn more or less than delivering to another port. Therefore, we associate to each port i a revenue $R_i > 0$ (in dollars) which express the total earned by the operator when a vessel serves the port. Analogously, we associate a penalty $\alpha_i \geqslant 0$ (in dollars) to be paid by the shipping operator if it decides to skip service at the port. This penalty can be also used to model the cost of outsourcing the service.

Figure 2a shows a simple instance of the problem, with one hub and two ports. In Figure 2b we create one node for each element of P' and draw solid arcs to represent that a ship can sail from any port to any other

port. The dotted arcs represent a ship that, at the same port, first performs a delivery and then a pickup.

3.1. Graphs

For each vessel v we create a space-time graph $G_v = (N, A_v)$. The node set is common to all graphs and is defined as $N = \{\sigma, \tau\} \cup P' \times T$ where σ and τ are, respectively, a source and sink node. All other nodes (i,t) represent a port at a specific time instant; G_v is, therefore, a time-expanded graph. The arcs A_v can be partitioned in four sets:

- Starting arcs of type $(\sigma, (0,t))$, that link the source node with a node representing the hub.
- Ending arcs of type $((0,t), \tau)$, that link a node representing the hub with the sink node.
- Delivery-to-pickup arcs of type $((i,t),(i+n,t'))$, that link the delivery and pickup operations at the same physical port.
- Sailing arcs of the type $((i,t),(j,t'))$, with $j \neq i+n$ if $i, j \in P$, that represent the sailing from port $i \in P'$ to port $j \in P'$.

The time used for the operations at the destination node of the arc is precomputed and included in the arc itself. To understand how this is done, consider the example of a sailing arc from port $i \in P$ to port $j \in P$, modelling the sailing of vessel $v \in V$, starting at time t and sailing at speed $\omega_k^v \in \Omega^v$. Assume that the draught limits are respected ($m_i^{\mathrm{p}} \geqslant m_v^{\mathrm{v}}$ and $m_j^{\mathrm{p}} \geqslant m_v^{\mathrm{v}}$), that i is open at time t, and that visiting j immediately after i does not violate capacity constraints ($p_i + p_j < Q_v$ if $i, j \in P^+$, $d_i + d_j < Q_v$ if $i, j \in P^-$, $p_i + d_j < Q_v$ if $i \in P^+, j \in P^-$). If these conditions do not hold, then the arc is not created.

The arrival time instant of vessel v will be $t_{\mathrm{arr}} = t + \lceil \Delta_{ij} / \omega_k^v \rceil$. Let t' be the time instant in which the handling operations at j can be completed; if t_{arr} corresponds to a time when port j is open, then $t' = t_{\mathrm{arr}} + h_j$, otherwise a waiting time has to be factored in, as the ship cannot berth or be operated on, while the port is closed. We can then create the arc $((i,t),(j,t'))$.

This process is repeated for each pair of ports, for each vessel $v \in V$, and for each speed $\omega \in \Omega^v$. The delivery-to-pickup arcs are created similarly, while the starting and ending arcs are created for all nodes representing the hub. A starting arc of type $(\sigma, (0,t))$ means that the vessel is leaving the hub at time t and therefore has been waiting idly at the hub during the time period $[0,t]$. Analogously, an ending arc of type $((0,t),\tau)$ represents a ship concluding its rotation at time t and waiting idly during time period $[t+1, m]$.

Notice that all costs, as well as the revenue, can be modelled on the arcs. For example, the cost of a sailing arc $a = ((i,t),(j,t'))$ is set to:

$$c_a = \Gamma_v^{\mathrm{TC}}(t'-t) + \Gamma_v^{\mathrm{H}}(t' - t_{\mathrm{arr}}) + \Gamma_{vk}^{\mathrm{B}}(t_{\mathrm{arr}} - t) + \Gamma_{vj}^{\mathrm{M}} + \Gamma_j^{\mathrm{PF}} + \Gamma_{vj}^{\mathrm{PV}} - R_j \tag{1}$$

The costs related to idle waiting times in starting and ending arcs are similarly incorporated into the arc cost.

Figure 2c shows the time-expanded version of the graph depicted in Figure 2b, for a particular vessel (for simplicity, we omit vertices σ, τ). In the figure, we only draw two arcs: they represent a ship sailing from "Port 1 Pickup" to "Port 2 Pickup". We can imagine, for example, that the first arc represents the ship leaving Port 1 at time 1, arriving at Port 2 at time 2, and finishing the loading operations at time 3. The second arc again models leaving Port 1 at time 1 but, sailing more slowly, Port 2 is reached only at time 3, and the loading operations are completed at time 4.

3.2. Integer formulation

We define a route for vessel v as a succession of consecutive arcs in A_v such that the first starts at σ and the last ends at node τ. If every port $i \in P$ is visited at most once, and if the capacity constraint is not violated,

we say that the route is feasible. The set of feasible routes for vessel v is denoted as R_v and the set of all feasible routes as $R = \bigcup_{v \in V} R_v$. Notice that it is possible to assign a cost c_r to each route r by summing the costs of the arcs that compose the route. Finally, let $\varepsilon_{ri} \in \{0, 1\}$ be a parameter with value 1 iff port i is visited by route r. We can formulate a model for the FNDP by considering binary variables x_r taking value 1 iff route $r \in R$ is part of the solution:

$$\text{(MP)} \qquad \min \sum_{r \in R} c_r x_r + \sum_{i \in P} \alpha_i \left(1 - \sum_{r \in R} \varepsilon_{ri} x_r\right) \tag{2}$$

$$\text{s.t.} \sum_{r \in R} \varepsilon_{ri} x_r \leqslant 1 \qquad \forall i \in P \tag{3}$$

$$\sum_{r \in R_v} x_r \leqslant 1 \qquad \forall v \in V \tag{4}$$

$$x_r \in \{0, 1\} \qquad \forall r \in R \tag{5}$$

The objective function (2) minimises the total cost of selected routes and the penalties paid. (A solution yields a profit if the value of the objective function is negative.) Constraint (3) ensures that each port is visited by at most one vessel, constraint (4) guarantees that we do not use more vessels than available, and finally constraint (5) specifies the variables' domain. Notice that, unlike problems in the VRP family, where the requirement that all customers need be served leads to set-covering $\geqslant$-constraints, our model has $\leqslant$-constraints in (3).

Problem (MP) is called the *master problem*. An obvious drawback of (MP) is that sets R_v are too large to enumerate in practice, since the number of feasible routes grows exponentially with the size of the graphs. We then consider a version of (MP) where the sets R_v are substituted with much smaller sets R'_v, and the integrality requirement on the variables is dropped. This new problem is called the *restricted master problem* (RMP). The idea behind the branch-and-price algorithm we propose is to solve (MP) by branch-and-bound: at each node the linear programme (RMP) is solved and new columns are added to R'_v. We search promising columns to add, by solving a *pricing subproblem* $(\text{SP})_v$ for each vessel v. If the subproblem produces new columns with negative reduced cost, from dual theory, we know that the objective function of (RMP) will improve when they enter the base (but for degenerate cases). If, on the other hand, the subproblems proves that no negative reduced cost column exists, (RMP) has been solved to optimality and the node is considered explored. Let $\pi_i \geqslant 0$ be the dual variables of (3) and $\mu_v \geqslant 0$ be the dual variables of (4). Notice that the objective function can be rewritten as

$$\sum_{r \in R} \left(c_r - \sum_{i \in P} \alpha_i \varepsilon_{ri}\right) x_r + \sum_{i \in P} \alpha_i \tag{6}$$

and the last sum is a constant. Therefore, the dual cost of a column corresponding to route $r \in R_v$ is:

$$\hat{c}_r = c_r + \sum_{i \in P} (\pi_i - \alpha_i) \varepsilon_{ri} + \mu_v \tag{7}$$

The dual cost is given by the original route cost; then, for each visited port $i \in P$, the dual price π_i is added and the corresponding penalty α_i is removed; finally, a dual price μ_v is paid, only depending on the used vessel $v \in V$.

4. Solution of the pricing subproblem

The pricing subproblem $(\text{SP})_v$ is a shortest path problem with resource constraints (SPPRC), as routes for vessel v are paths in G_v from the source node σ to the sink node τ, and the resource constraints are used to ensure the routes' feasibility. In some sense, this problem can also be seen as an elementary SPPRC

(ESPPRC). Graphs G_v are acyclic, and therefore any path would be elementary. In our case, however, we require that every port $i \in P$ is visited at most once, i.e. that every subset of nodes $N_i = \{(i,t) \mid t \in T\}$ has at most one inbound and one outbound arc. In the rest of this paper we will refer to this property when we speak of elementariness.

Notice that the elementariness requirement can be dropped, and routes that visit the same port multiple times can be generated, as long as the corresponding columns are then removed by suitable branching rules in the master problem (as we will explain in Section 5.3). For this reason, in this section we propose algorithms to solve both the ESPPRC and the SPPRC.

4.1. Greedy-randomised heuristic for the ESPPRC

We can attempt to find negative-reduced-cost elementary path with a simple greedy algorithm that builds a path starting in σ and then proceeds by choosing one random arc among the K_1 outgoing arcs of least reduced cost that do not close a cycle, until τ is reached. The same procedure can also be applied backward, starting in τ and ending at σ. After the path is constructed, we can check that the capacity constraint has not been violated by a single pass over the list of visited ports; if the path is not capacity-feasible, or if its reduced cost is non-negative, it is discarded. The algorithm can be applied in both directions many times, as its computational time is very small, thereby increasing the chances that a feasible path of negative reduced cost is found.

4.2. Exact dynamic programming algorithm for the ESPPRC

The ESPPRC can be solved exactly via dynamic programming by using a label-setting algorithm (see, e.g., Irnich and Desaulniers [20] for a review on solution methods for shortest path problems with resource constraints). We associate a label L to each partial path from σ to a node (i,t). The label components are similar to those introduced by Dell'Amico et al. [15] in the context of a Vehicle Routing Problem with simultaneous distribution and collection:

- $\eta \in N$, the current node (i,t) the path is visiting.
- $\Pi \in \mathbb{N}$, the amount of cargo that vessel v can pick up after visiting the current port.
- $\Delta \in \mathbb{N}$, the amount of cargo that vessel v can deliver after visiting the current port.
- $\vec{V} \in \{0,1\}^P$, a binary vector whose j-th component is 0 iff port $j \in P$ can be visited at some point after the current port. This vector is used to keep track of visited ports so to ensure that the route is elementary.
- $C \in \mathbb{R}$, the cost associated with the partial path. This is given by the sum of the costs of the arcs traversed, plus the prices π_j paid and minus the penalties α_i avoided at visited ports. If $\eta = \tau$ we also add the dual price μ_v.

An initial label is created with $\eta = \sigma$, $\Pi = \Delta = Q_v$, $\vec{V} = \vec{0}$, and $C = 0$. Notice that a label L can then be thought of as an element of the space $S = N \times \{0,\dots,Q_v\}^2 \times \{0,1\}^{|P|}$ equipped with a cost function $C : S \to \mathbb{R}$.

In the following we will use the convenient convention that $p_i = 0$ for all $i \in P^-$ and $d_j = 0$ for all $j \in P^+$. Let us describe how to update a label L associated with a partial path to (i,t) when the path is extended to a node (j,t'), with $j \in P$, along a sailing arc. We will call the new label $L' = (\eta', \Pi', \Delta', \vec{V}')$ and its cost C'. First of all we check if such an extension is feasible. This is the case if there is an arc a connecting (i,t) with (j,t'), if j is a visitable port (i.e. the j-th component of $\vec{V}$ is 0) and if there is enough available capacity on the vessel (i.e. $\Pi \geqslant p_j$ and $\Delta \geqslant d_j$). If the extension is feasible, we set the components of new label L' as follows: component η' will be set to (j,t'); $\vec{V}'$ will be equal to $\vec{V}$ except for the j-th coordinate, that will be set to 1, as it is not possible to visit port j again; the cost component will be updated as

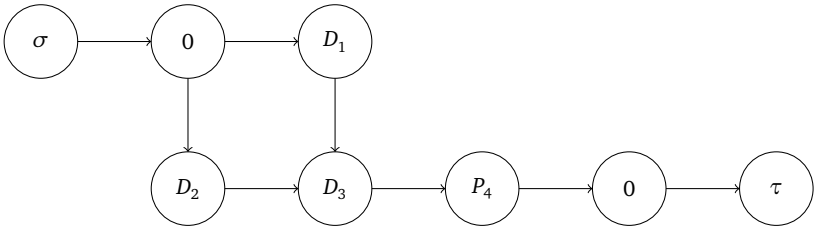


Figure 3: Example graph that shows the importance of correctly ordering the relaxed labels.

$C' = C + c_{av} + \pi_j - \alpha_j$. Let us now consider component Π': since collecting cargo consumes the associated resource "amount of load that can be picked up after visiting j" of a number of units equal to the amount of cargo we collect, while delivering it does not, it will be updated as $\Pi' = \Pi - p_j$. Component Δ', on the other hand, represent the resource "amount of load that can be delivered after visiting j" and both a pickup and a delivery operation reduce this resource by a number of units equal to the amount of cargo picked up or delivered. Therefore, it is updated as $\Delta' = \min\{\Pi - p_j, \Delta - d_j\}$.

Similar rules are followed when extending along other type of arcs. In particular, an extension to the hub $(0, t)$ is always possible, and will have the effect of resetting components Π and Δ to their original value of Q_v. Finally, we charge the dual cost μ_v associated with the vessel on the arcs used to reach the sink (i.e. on the "ending arcs" described in Section 3.1); the cost of the corresponding label will be updated as $C' = C + \mu_v$.

We now define a dominance criterion to compare two labels $L_1 = (\eta, \Pi_1, \Delta_1, \vec{V}_1, C_1)$ and $L_2 = (\eta, \Pi_2, \Delta_2, \vec{V}_2, C_2)$ representing partial paths up to the same node. We say that label L_1 dominates L_2, and we write $L_1 \prec L_2$, if: (a) $C_1 \leqslant C_2$; (b) $\vec{V}_1 \leqslant \vec{V}_2$ component-wise; (c) $\Delta_1 \geqslant \Delta_2$; (d) $\Pi_1 \geqslant \Pi_2$; (e) at least one of the previous inequalities is strict.

4.3. Exact dynamic programming algorithm for the SPPRC

Since the ESPPRC is an $\mathcal{NP}$-hard problem, we can consider the labelling algorithm on a relaxed state space (state here is a synonym for label). We project the state space S into a lower-dimensional space $S' = N \times \{0, \dots, Q_v\}^2$. The projection function sends an element $L = (\eta, \Pi, \Delta, \vec{V}) \in S$ to the element $L' := \text{proj}(L) = (\eta, \Pi, \Delta) \in S'$. As shown by Christofides et al. [14], the interesting property of state space relaxation is that the cost of the new label is $C(L') \leqslant \min_{L \in \text{proj}^{-1}(L')} C(L)$, and therefore provides a lower bound on the minimum cost of the corresponding label in the original space. The drawback is that S' contains labels that would be infeasible for the ESPPRC and, in particular, there are labels that correspond to paths that are not elementary. Since we only have forward arcs in the time-expanded graph, however, eventual cycles will be finite. The state-space-relaxed labelling algorithm, therefore, solves the SPPRC. Since $|S'| = |N| \cdot (Q_v + 1)^2$, the algorithm has pseudo-polynomial complexity. The extension function of this algorithm will omit the missing component and the dominance criterion will omit condition (b). We will denote the problem solved by the state space relaxed version of our labelling algorithm as (SP')$_v$.

Label extension order. It is important to observe that the fundamental assumption of label-setting algorithms (if a label L_1 dominates a label L_2, then all extensions of L_1 dominate the corresponding extensions of L_2) is valid for the relaxed state space, only if we process the labels in a particular order. To see why this is the case, consider the graph in Figure 3 where, for simplicity, we omitted the time component and all arcs have cost 1. Ports D_1, D_2, D_3 are delivery ports with demands, respectively, 5, 4 and 5. Port P_4 is a pickup port with demand 10. The vessel's capacity is 10. Consider two paths: $r_1 = (\sigma, 0, D_1, D_3, P_4, 0, \tau)$ and $r_2 = (\sigma, 0, D_2, D_3, P_4, 0, \tau)$. The labels associated respectively with these two paths at node D_3 are $L_1 = (D_3, 10, 0)$ and $L_2 = (D_3, 10, 1)$. Having both the same cost, we should conclude that L_2 dominates L_1. However, if we consider their extensions to port P_4, we see that both labels would be extended into

$L'_1 = L'_2 = (P_4, 0, 0)$. Therefore it is not true that any extension of L_2 dominates the corresponding extension of L_1. This observation also applies to the state relaxation used in Dell'Amico et al. [15], where the authors projected component $\vec{V}$ into $\sum_{i \in P} \vec{V}_i$, as both partial paths in D_3 have visited the same number of ports. To overcome this limitation, one has to avoid extending a label that could later be removed by domination; otherwise, the child label would become orphaned. This can be achieved by appropriately sorting the labels and processing them in order. In our case, we equip S with a suitable total order relation, and make sure that a label is not extended before all its predecessors are. The order we use in our algorithm is a lexicographic comparison, starting with component Δ (lowest first), followed by Π (lowest first) and finally by the cost (highest first).

4.4. Acceleration techniques

Both labelling algorithms for the ESPPRC and the SPPRC can be accelerated heuristically, by pruning the time-expanded graph. We propose two strategies for arc removal. The first strategy involves sorting all arcs in each graph by their reduced cost, and only keep the $K_2 \cdot |A_v|$ ones with lowest reduced cost, where $K_2 \in (0, 1)$ is a parameter. This sparsification method considers both the original cost of the arcs and the reduced prices to pay at the ports; we refer to this technique as (C+P).

The second strategy, instead, sorts the ports by their respective reduced prices π_i. Let $\bar{\pi}$ and $\underline{\pi}$ the highest and lowest, respectively, reduced prizes (the lowest one being the most desirable), and $\bar{i}$ and $\underline{i}$ the ports corresponding, respectively, to $\bar{\pi}$ and $\underline{\pi}$. Then each arc is removed with a probability directly proportional to the dual price of its target port, with arcs incoming to $\underline{i}$ having probability 0, and arcs incoming to $\bar{i}$ having probability K_3, where $K_3 \in (0, 1]$ is a parameter. This sparsification method only takes into account the reduced prices, independently from the original cost of the arcs; we refer to this technique as (P).

5. Branch-and-price algorithm

As mentioned in Section 3, we solve the FNDP with a branch-and-price algorithm. We explore the branch-and-bound tree according to a best-first strategy: when a node is explored, the next node is the one whose parent's lower bound is the highest (with ties broken randomly).

5.1. Column generation

At each node of the tree we solve (RMP) and use its dual values to find negative reduced cost columns, by solving k subproblems $(\text{SP'})_v$. We solve the subproblem using both the heuristics and the exact methods presented in Section 4. Heuristic pricing has been successfully employed in order to speed up column generation in a variety of routing problems (see, e.g., Dumas et al. [18], Savelsbergh and Sol [33], and Desaulniers et al. [16]).

The following methods are used sequentially; if any of them finds a route of negative reduced cost, the column generation phase is halted and (RMP) is re-solved.

1. The first heuristic is the randomised-greedy heuristic of Section 4.1, which is tried 100 times in each direction, with parameter $K_1 = 10$. This heuristic is very quick and has the advantage of producing feasible columns; however, it is usually able to produce negative-reduced-cost columns only at the beginning of the exploration of a branch-and-bound node.
2. Next, we solve the SPPRC by dynamic programming, employing the acceleration techniques presented in Section 4.4. We use (C+P) with parameter $K_2 = 0.25$ and (P) with parameter $K_3 = 1$.
3. Finally, we solve the SPPRC by dynamic programming, on the complete graph.

5.2. Column management

We solve the first iteration at the root node with a pool made of one single dummy column, corresponding to a visit to each port. We attribute a very high cost to this column and consider as infeasible any solution that includes it among the base columns. If unused vessels can be chartered out, and β_v is the revenue generated by chartering vessel $v \in V$, another possibility is to initialise the column pool with one column for each vessel, each with cost $-\beta_v$ and not visiting any port.

At each node we remove duplicate columns from the column pool. Notice that neither the order in which ports are visited, nor the visit times, nor the visits to the hub are encoded in the columns, so it is possible to have duplicate columns with different costs. In this case we only keep the column with the lowest cost.

5.3. Branching

In classic VRP-like problems, the integrality and feasibility of the solution is often imposed by considering a route corresponding to a fractional column, and a customer (say i) visited by that route, which is also covered by another fractional column, such that the two customers (say j, k) preceding i in the two routes differ. Branching is performed by imposing in one child node that arc (j, i) is not used (e.g. by removing that arc from the graph), and in the other child that node i can only be reached via j (e.g. by removing all other arcs leading to i and all other arcs leaving j). In this latter node, because of the set covering constraints, nodes i and j need to be visited, and therefore the use of arc (j, i) is guaranteed.

Since in our model (2)–(4) we do not have such set covering constraints, this branching rule would not actually guarantee an integer solution in the second node. In our problem, in fact, more decisions need to be made: whether a port is visited or not; if so, by which vessel; once the vessel is fixed, by using which arc.

For this reason, we propose the branching rules described below. The application of these branching rules will also guarantee that infeasible columns (i.e., those corresponding to routes that visit a port more than once) will not appear in the optimal solution. The order in which the following rules are applied is: (1) branch on port visit; (2) branch on vessel; (3) branch on successive visits; (4) branch on arc selection.

5.3.1. Branching on port visit

The first branching rule is used to determine whether a port is visited, by any vessel. This branching rule was also used by Boussier et al. [7] in their branch-and-price algorithm for the TOP.

Let, for a given solution $\hat{x}$ of (RMP) and for a port $i \in P$, $x(i) = \sum_{r \in R'} \varepsilon_{ri} \hat{x}_r$ (R' is the set of routes active at the node). If not all values $x(i)$ are integer (i.e., either 0 or 1) then one port is visited with fractional flow. We then select the port i for which the quantity $x(i)$ is most fractional (i.e., closer to 0.5), and create two branches.

In the first branch i is visited and, in the master problem, constraint (3) is replaced by

$$\sum_{r \in R} \varepsilon_{ri} x_r = 1 \tag{8}$$

The subproblem remains unchanged, but notice that now the dual variable π_i associated with the port is unrestricted, meaning that a path visiting i will now potentially collect a prize $\pi_i \leqslant 0$ which will be deducted from the cost of the partial path.

In the second branch, i is not visited. In the master problem, we remove all columns corresponding to routes r for which $\varepsilon_{ri} = 1$. In the subproblem, we remove the nodes of set N_i from all graphs. Therefore, no column covering i will appear in the subtree of this branch.

5.3.2. Branching on vessel selection

If all ports have an integer value for $x(i)$, it can still happen that there is a port i which is visited by more than one vessel. Define, for a solution $\hat{x}$ of (RMP), a port $i \in P$, and a vessel $v \in V$, the quantity $x_v(i) = \sum_{r \in R'_v} \varepsilon_{ri} \hat{x}_r$ (R'_v is the set of routes associated with vessel v, and active at the node). We then select the port i and the vessel v for which the quantity $x_v(i)$ is most fractional, and create two branches.

In the first branch, we want i to be visited by v (if it is visited at all). In the master problem, we remove all columns of $r \in R'_w$ with $\varepsilon_{ri} = 1$, for all vessels $w \neq v$. In the subproblem, we remove all nodes of N_i from all graphs associated with vessels $w \neq v$. Notice that if the inequality (3) corresponding to i was already transformed into an equality by the previous branching rule, then we can also remove all columns of $r \in R'_v$ with $\varepsilon_{ri} = 0$, as in this case we know that i will be visited, and will be visited by v.

In the second branch, i is not visited by v. Therefore, in the master problem, we remove all columns of R'_v which cover i. In the subproblem, we remove all nodes of N_i from the graph associated with v.

5.3.3. Branching on arc selection

Consider a solution which does not trigger any of the preceding branching rules, i.e. each port has integer flow, and is visited by only one vessel. Let $r \in R'$ be the route corresponding to the most fractional column in the base, and v the associated vessel.

A branching rule sufficient to ensure that the final solution be integer and without infeasible columns would consist in considering any arc a of r connecting two ports, say linking (i,t) to (j,t') in the time-expanded graph, and creating two branches.

In the first branch, we force v to use arc a if travelling from i to j. To do so, all columns corresponding to routes $r \in R'_v$ such that $\varepsilon_{ri} = \varepsilon_{rj} = 1$, but which do not use a, are discarded from the master problem. In the subproblem level, we remove all other arcs linking nodes of N_i with nodes of N_j from G_v.

In the second branch, we forbid the use of the arc; in the master problem we remove the routes of R'_v which use a, and in the subproblem we simply remove the arc from the graph G_v.

Notice that this rule can also exclude routes containing cycles, as in these routes there is at least one port which is visited twice, i.e. by using two incoming arcs, and one of the arcs can be forbidden by using this rule. In this case, the rule need be applied to integer columns as well, as a route containing a cycle could be selected with coefficient 1.

However, the rule produces two unbalanced subtrees, as one branch imposes a much stronger condition than the other. Furthermore, routes where v does not visit neither i nor j can appear in both branches. To improve the convergence of the algorithm we also devise another branching rule, described in the next subsection.

Notice, finally, that this branching rule used in conjunction with the previous two forms a complete branching scheme. Indeed, in the branch forbidding the use of the arc, its flow will be obviously integer (in particular, it will be 0). When imposing the use of the arc, however, the optimal solution to (RMP) could still give fractional flow. In this case, however, if the total flow on port j is fractional, we will perform branching on the port visit. In the branch that excludes j the flow on the arc will be again obviously integer (and equal to 0). In the other branch, we still have the possibility that the flow be fractional. But in this case, since the total flow on j is 1, there must be another route associated with a different vessel whose corresponding column is also in the base. In this case we will perform branching on vessel selection. In the branch that assigns j to v the flow on the arc is forced to be 1; in the other branch, it is forced to be 0.

5.3.4. Branching on successive visits

Let $r \in R'_v$ be the route corresponding to the most fractional column in the base. If there is another fractional column corresponding to a route $r' \in R'_v$ ($r' \neq r$), such that there is a port i visited by both r and r', and the ports preceding i in r and r' are different (say, j and j'), then we can perform binary branching by respectively forcing and forbidding the consecutive visit of ports j, i. This rule can again be applied to routes corresponding to integer columns, in order to remove cycles.

In the first branch, all columns of R'_v whose corresponding route does not visit the succession of ports j, i are discarded from the master problem; regarding the subproblem, in G_v we remove all arcs from nodes of N_j to nodes of N_k (for $k \neq i$), and from nodes of N_k to nodes of N_i (for $k \neq j$). In the second branch, all columns of R'_v whose corresponding route visits j, i in succession are discarded from the master problem; regarding the subproblem, in G_v we remove all arcs from nodes of N_j to nodes of N_i.

5.4. Upper bounding

To strengthen the upper bound, at the end of the exploration of the root node, the master problem is solved as an integer problem with the feasible columns currently in the column pool, using a black-box commercial solver. This step is performed only if the column pool has fewer than 1500 columns.

6. Results

In this section we describe how the test instances were generated starting from instances already present in the literature, and we provide computational results that highlight both the performance of our algorithm and key features of optimal routes.

6.1. Instance generation

The instances used in this paper are derived from the library LinerLib, presented by Brouer et al. [8]. We considered the two LinerLib scenarios *Baltic* and *Western Africa* (*WAF*). Both scenarios include a single hub: Bremerhaven for the Baltic scenario and Algeciras for the Western African one. In the Baltic scenario, we have a 1-week planning horizon, modelled with a discretisation of 2 hours per time interval; in the WAF scenario, the planning horizon is of 4 weeks, and each time instant represents 8 hours. The Baltic scenario comprises 13 ports (see Figure 4a) and 6 vessels, while the Western African one has 20 ports (see Figure 4b) and 10 vessels. Since port NGAPP in the WAF scenario has a delivery demand which is higher than the capacity of the largest ship (1573 TEU vs. 800 TEU), we duplicated that port. Port NGAPP-1 has been given a delivery demand of 800 TEU and the whole pickup demand, while port NGAPP-2 has been given the remaining delivery demand (773 TEU). This effectively brings the number of ports in the WAF scenario to 21.

In both scenarios, the bunker price is of \$375/tn, and all α_i have been set to 0, thereby considering the pure opportunity cost of each service (revenue minus cost of performing the service). Furthermore, each vessel can sail at a low, medium, or high speed (the minimum and maximum speeds are obtained from the LinerLib data).

We generated respectively 12 and 8 base instances for the Baltic and WAF scenarios. These instances share the same network topology of the LinerLib original instances, but have different values for time windows, transit times, and handling times:

- We considered instances both with and without closing time windows. When they are present, their centres are distributed evenly along the time horizon, to simulate ports closing at night, each day of the week. For the Baltic scenario, we have three options: no time window; all time windows have duration of 1 time interval; time windows have durations of either 1 or 3 time intervals (the actual

value is chosen at random for each port). For the WAF scenario, we only have two options: no time window, or all time windows with duration 1.

- We also generated instances with and without maximum transit times. When the transit times were enabled, their value was comprised between 5 and 6.5 days for the Baltic scenario, and between 23 and 27 days for the WAF scenario. The actual transit time for each port was chosen according to a uniform random distribution over said intervals.
- The handling times were also distributed in intervals ($[2,4]$ or $[3,5]$ time units for the Baltic scenario; $[1,1]$ or $[1,2]$ time units for the WAF scenario). However, the actual value for each port has been chosen proportionally to the number of containers to be handled at the port.

We thus generated a total of 20 base instances (available at Santini [32] together with the solver code) which we use in Section 6.2 to verify that the proposed branch-and-price algorithm attains state-of-the-art performances in terms of solution time and quality vs number of ports and vessels considered. We then generated further instances, which were used for the scenario analysis presented in Section 6.3. The objective of the scenario analysis is to assess the impact of external conditions, designer's decisions, and modelling precision on the optimal solutions. These instances were based on the Baltic scenario. Typical questions that are investigated in the scenario analysis are, e.g.: under which conditions it is convenient to buy capacity on competitors? what happens when there is a surge in bunker prices? will considering more possible speeds in the model lead to significantly better solutions? will longer routes lead to better utilisation of ship capacities?

In order to perform this analysis, we varied further characteristics of the base instances:

Bunker price While the bunker price was fixed to an average value of \$375/tn in the base instances, in this scenario analysis we use the values $\{250, 300, 375, 450, 500\}$ to assess what changes in the solution under low, medium, and high bunker prices. This price is in practice subject to changes for two main reasons: the volatility of crude oil price, and the possibility that certain countries will pass legislation forcing the use of low-sulfur bunkers, which are less polluting but much more expensive.

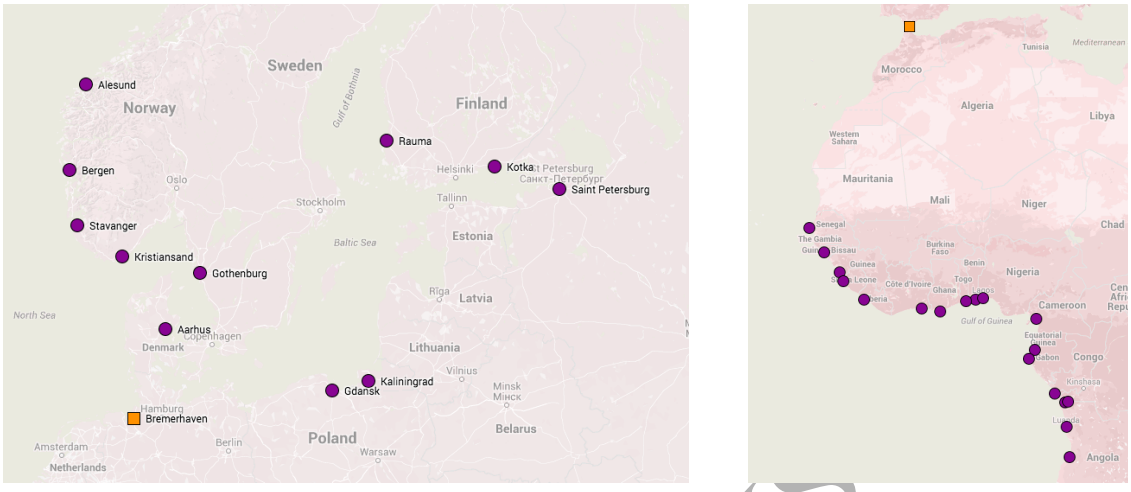
Speeds In this analysis we use 1, 3 (base instances), and 5 possible speed values for each vessel, in order to check whether the increased complexity (especially in terms of arcs being created in the graphs G_v) given by adding more speed values leads to significantly better solutions.

Demand In this analysis we consider scenarios with demand multiplied by a factor of 1.0 (base instances), 0.8, 0.6, and 0.4. This analysis is particularly important, as one of the major consequences of a global financial and consumption crisis is a steep decline in the volumes of goods traded and shipped by sea.

Penalties We use penalties to model the purchase of capacity on a competitor. In particular, while in the base instances we only had opportunity costs, in this scenario analysis we can outsource a service. In this case, we can still earn a small percentage of the revenue (1%, 5%, or 10%), while most of it is transferred to the competitor.

Time horizon We solved the Baltic instances with time horizons of 1 (base instances), 2, and 3 weeks. The number of hours per time interval were, respectively, 2, 4, and 6 for WAF. The number of ships available is scaled according to the time horizon (1/2 and 1/3 of the original fleet size, respectively) to reflect the fact that multiple ships need to sail the same rotation at once. This simplification was possible due to the fact that ships in LinerLib are grouped by "vessel class" and ships belonging to the same vessel class are considered identical. It is, therefore, possible to reduce the number of vessels by scaling each vessel class separately.

In total, we generated additional 48 instances for the bunker price analysis, 24 for the speeds analysis, 36 for the demand analysis, 36 for the penalties analysis, and 24 for the time horizon analysis.



(a) The Baltic scenario.

(b) The WAF scenario.

Figure 4: Ports in the considered scenarios. Map data: Google.

Instance	Time	SP Time %	Gap %	RGap %	Cols	NNodes	Depth	Ports	HLE
Baltic1	0.77	96.75	0.00	0.00	72	1	1	2.5	0.84
Baltic2	4.07	99.54	0.00	0.00	91	1	1	3	0.82
Baltic3	13.87	98.67	0.00	0.00	145	1	1	3	0.82
Baltic4	9.39	98.65	0.00	0.00	113	1	1	3	0.82
Baltic5	8.66	99.16	0.00	0.00	114	1	1	3	0.82
Baltic6	4.99	99.60	0.00	0.00	88	1	1	3	0.83
Baltic7	11.67	99.08	0.00	0.00	135	1	1	3	0.82
Baltic8	1.71	98.80	0.00	0.00	64	1	1	2.5	0.84
Baltic9	1.65	98.18	0.00	0.00	70	1	1	2.5	0.84
Baltic10	2.21	96.39	0.00	0.00	78	1	1	2.5	0.86
Baltic11	4.08	98.25	0.00	0.00	95	1	1	2.5	0.84
Baltic12	2.44	98.62	0.00	0.00	75	1	1	2.5	0.84
WAF1	3600.00	91.66	0.04	0.34	3402	321	151	3.25	0.40
WAF2	819.04	93.40	0.00	0.34	5132	45	13	3.5	0.40
WAF3	1382.66	93.93	0.00	0.71	5036	89	16	3.38	0.41
WAF4	1229.08	94.98	0.00	0.34	5396	47	12	3.38	0.40
WAF5	555.15	92.61	0.00	0.21	4255	25	9	3.63	0.40
WAF6	319.76	93.38	0.00	0.21	3474	17	8	3.5	0.38
WAF7	275.23	87.44	0.00	0.21	3101	11	5	3.5	0.40
WAF8	497.11	92.99	0.00	0.21	4754	35	13	3.38	0.41

Table 2: Computational results for the base instances.

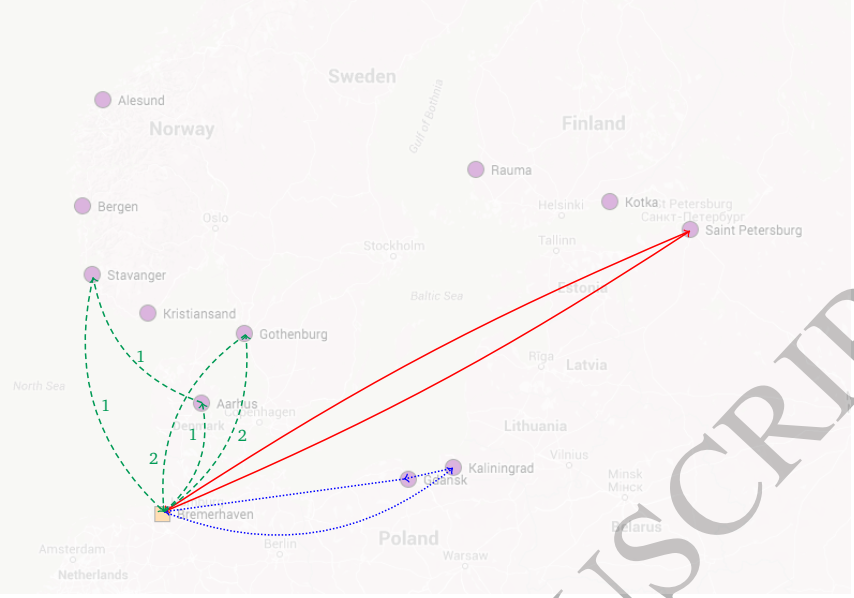


Figure 5: Optimal solution for instance Baltic1.

6.2. *Computational results*

Computational experiments were run on a dual-core 3.10GHz Xeon CPU with 4GB of RAM. The LPs were solved using CPLEX 12.6, limited to the use of one thread. We set a time limit of one hour. Table 2 reports the results for the Baltic and Western African scenarios.

Column "Time" reports the execution time in seconds, while column "SPTime" tells the time, in percentage, spent solving the subproblem. Columns "RGap" and "Gap" report the optimality gap percentage, respectively at the root node and at the end of the solution process. The gap is calculated as $100(\text{UB} - \text{LB})/\text{UB}$. Column "Cols" indicates the size of the column pool, "NNodes" gives the number of branch-and-bound nodes explored, and "Depth" is the maximum depth reached in the branch-and-bound tree. Column "Ports" reports the average number of ports, excluding the hub, visited by each vessel (pickup and delivery ports are counted separately). Finally, column "HLE" lists the highest load efficiency, a metric of capacity utilisation described in Section 6.3.

Table 2 shows that the proposed approach is able to solve to optimality (or almost to optimality) instances of realistic size. Our results are in line or better with recent work in maritime optimisation: Reinhardt and Pisinger [31], e.g., solve instances with up to 15 ports; Plum et al. [27] solve instances with up to 25 ports, but only consider one vessel. Also notice that, since the decisions for pickup and delivery are independent, each port we considered (but for the hub) is modelled twice, thus we deal with graphs of 25 (Baltic) and 39 (WAF) nodes, before time expansion.

The root node gaps show that the linear relaxation of (MP) is very strong. In particular, comparing with an earlier version of this paper, we noticed that relaxing the requirement that each route visits the hub exactly once provides both a stronger dual bound at the root node, and quicker convergence towards optimality. This comes at no cost in terms of practical solution quality, as the present version of the problem also produces routes which are closer to the real-life requirements of network planners.

All Baltic instances are solved at the root node, whereas branching is required for the WAF instances. Most of these instances are solved to optimality exploring fewer than 100 nodes; the only instance that requires exploring a large number of nodes and going deep in the branch-and-bound tree, is also the only open instance, WAF1. An analysis of the logs, furthermore, has shown that the branching rules described in

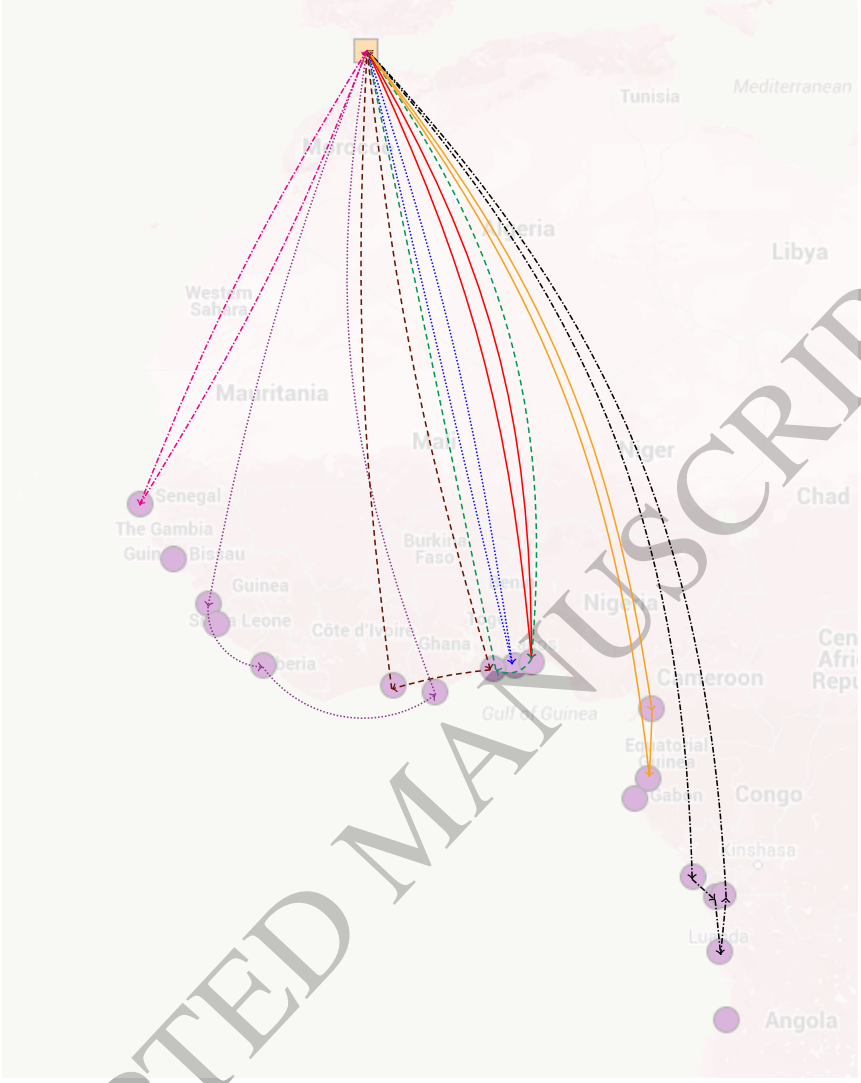


Figure 6: Optimal solution for instance WAF4.

Sections 5.3.1, 5.3.2 and 5.3.4 were sufficient to solve all closed instances to optimality, while the branching rule described in Section 5.3.3 was never used. Table 2 also shows that solving the linear programme (RMP) is computationally inexpensive, and instead almost all computing time is spent solving the subproblems.

We can also notice that, mostly due to time constraints, ship rotations are relatively short. For example, in the Baltic scenario, the highest average number of visited ports is 3. This raises the question if it would not be more efficient to simply enumerate all possible short routes. However, notice that we would have to take into account: the possible start times; the visits back to the hub; the possibility of performing deliveries, pickups, or both; the different speeds for each leg; the different vessel types. Even if we only consider a fraction of these possible combinations, an enumeration approach would be much more expensive than column generation even for the smaller instances, and definitely not feasible for the bigger ones.

Figures 5 and 6 show two optimal solutions, respectively to instances Baltic1 and WAF5. In Figure 5, for example, a butterfly route is used to serve first Aarhus and Stavanger and, after returning to the hub, Gothenburg. A dedicated service is performed in Saint Petersburg, which has high demand; on the other

Instance	Time		Cols	
	With	Without	With	Without
Baltic1	0.77	8.93	72	76
Baltic2	4.07	5.34	91	102
Baltic3	13.87	16.28	145	89
Baltic4	9.39	13.88	113	89
Baltic5	8.66	16.79	114	94
Baltic6	4.99	5.38	88	101
Baltic7	11.67	13.25	135	89
Baltic8	1.71	5.70	64	80
Baltic9	1.65	2.52	70	66
Baltic10	2.21	1.88	78	71
Baltic11	4.08	4.09	95	80
Baltic12	2.44	1.80	75	76
WAF1	3600.00	3600.00	3402	2327
WAF2	819.04	2851.12	5132	3598
WAF3	1382.66	3600.00	5036	4508
WAF4	1229.08	1480.49	5396	2466
WAF5	555.15	3600.00	4255	3895
WAF6	319.76	722.46	3474	1873
WAF7	275.23	323.51	3101	1690
WAF8	497.11	582.38	4754	2389
Average	437.18	842.79	1784.50	1187.95

Table 3: Solution times comparison with and without acceleration techniques.

hand, the port of Rauma (Finland) is distant and has very low demand, so its service would have a high cost and low profit, and the service is skipped. Some port might be unprofitable to serve, but an operator might still decide to serve it for prestige reasons or other commercial considerations. In this case, the network planner could direct the model in this sense, by applying an appropriate penalty α_i. This could happen, e.g., if the operator decided to serve more than just one port in Norway, by maintaining an additional presence in Bergen, Aalesund, or Kristiansand.

By analysing these solutions, we noticed two recurring features. First of all, there are ports which require dedicated services because of their high volume. Second, routes tend to serve ports clustered together, as is clear in Figure 6. These observations are validated by real-life practice where indeed some port is served by dedicated vessels and services are often separated into short-sea clusters and deep-sea clusters, as this configuration gives ports called on each type of service a relatively good transit time for both import and export volumes. Notice, e.g., the short-sea cluster Conarky – Monrovia – Takoradi served by the dotted purple route in Figure 6 and the deep-sea cluster Pointe-Noire – Boma – Lobito – Matadi served by the black dash-dotted line.

An analysis of the solutions of the base instances has shown an average vessel speed of around 15kn in the Baltic scenario, and 13.5kn in the Western African one, and that the cost of bunker accounts for around 30% and 20% of the total costs, respectively. These values are also in line with real-life data.

6.2.1. Subproblem Solution

In this subsection we report some considerations on the solution of the pricing subproblem. First of all, we note that the solution of the subproblem without state space relaxation has proven infeasible: for the Baltic instances we obtained an increase in computation times, but without any corresponding benefit, as

Value	Rev	SU	Srv	HLE	CTD	Spd	BC
250	+5.80	0.00	+5.30	-2.02	-0.58	0.69	-28.36
300	+3.36	0.00	+5.30	-2.02	-0.58	0.63	-16.98
375	0.00	0.00	0.00	0.00	0.00	0.00	0.00
450	-3.92	0.00	0.00	0.00	0.24	-0.03	11.72
500	-6.19	0.00	0.00	0.00	0.24	-1.13	23.28

Table 4: Scenario analysis for the "Bunker Price" value.

Value	Rev	SU	Srv	HLE	CTD	Spd	BC
1	-24.37	-20.83	-9.09	-43.51	-17.63	-19.94	-38.51
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	+0.36	0.00	+4.55	0.00	-0.81	0.21	-0.37

Table 5: Scenario analysis for the "Number of speeds" value.

Value	Rev	SU	Srv	HLE	CTD	Spd	BC
0.4	-78.75	-62.50	-36.36	-44.85	-22.44	-11.08	-25.26
0.6	-55.12	-25.00	+4.55	-56.42	-6.99	-8.17	-14.89
0.8	-8.52	0.00	+13.64	-16.22	+8.26	+0.31	-3.25
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.2	+6.67	-25.00	-16.67	-2.83	-13.74	-16.04	-40.99

Table 6: Scenario analysis for the "Demand" value.

all instances are already solved at the root node; for the Western African instances, we were not able to complete the column generation phase at the root node of any of them, when solving the ESPPRC.

Next, we investigated the impact of the acceleration techniques on the speed of the algorithm. Table 3 displays the total computation time (columns "Time") and the total number of columns generated (column "Cols") both with and without acceleration techniques. The computational time increases, on average, by 90% when we do not use acceleration techniques. Furthermore, we have two additional WAF instances which remain open after 1 hour. On the other hand, it is interesting to notice that the average number of columns generated is smaller, suggesting that column generation converges to the optimal solution of the relaxation in fewer iterations, but each iteration is more expensive.

6.3. Scenario Analysis

Tables 4 to 8 present the summary of the scenario analyses. Column "Value" reports the variations on the relevant value that is being changed (bunker price, number of speeds, demands, share of revenue kept when outsourcing, time horizon length). Each row aggregates, by taking the average, all instances which share the same relevant value. The underlined value is the one used for the base instances. All other columns report the percentage variation of some metric compared to its base value. If a metric has value m_v for one group of instances and value m_b for the base instances, the column will show the value $100(\frac{m_v}{m_b} - 1)$. The metrics considered are:

Rev The total revenue earned.

SU The total number of ships used. This value is adjusted appropriately when instances with different time horizons are considered.

Srv The total number of services performed.

HLE The highest load efficiency. For a given instance this value is the average of the highest load efficiency for each rotation. The highest load efficiency for a rotation is the highest quantity D/Q_v achieved at

Value	Rev	SU	Srv	HLE	CTD	Spd	BC
Keep 10%	+9.24	0.00	+1.52	-4.04	+0.48	-2.25	-4.88
Keep 5%	+4.60	0.00	+0.76	-2.02	+0.24	-1.13	-2.44
Keep 1%	+0.98	0.00	0.00	0.00	0.00	-0.04	0.00
Keep 0%	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 7: Scenario analysis for the "Penalties" value.

Value	Rev	SU	Srv	HLE	CTD	Spd	BC
1wk	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2wk	+18.70	+100.00	+27.27	+4.10	+42.73	-27.39	-18.90
3wk	+1.19	+100.00	+20.00	+4.58	+80.85	-23.10	-11.60

Table 8: Scenario analysis for the "Time horizon" value.

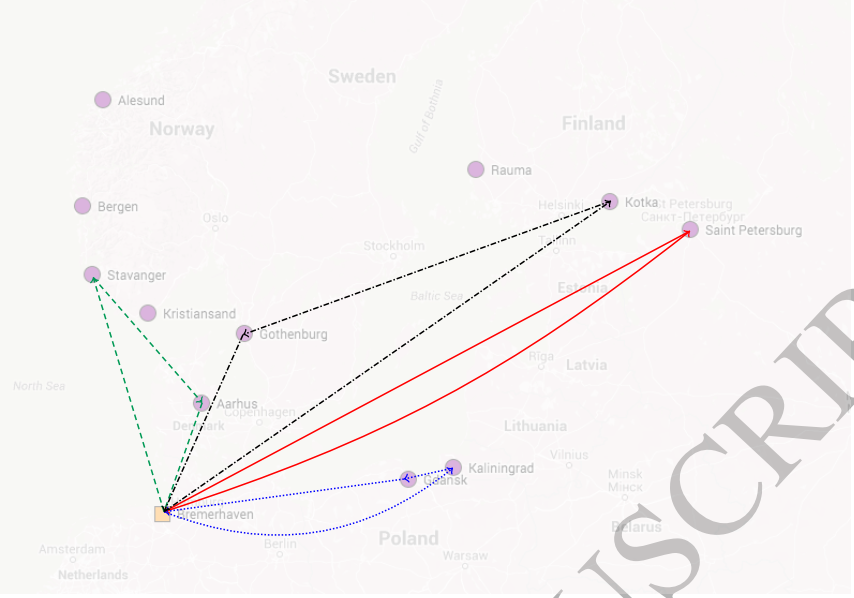


Figure 7: Optimal solution for the Baltic1 instance with demand coefficient of 0.8.

some point in the rotation, where D is the total quantity of cargo on board, and Q_v is the capacity of the considered vessel.

CTD The cargo travel distance. For a given instance this value is the average distance that each cargo has spent travelling on-board the ship.

Spd The average speed, considering in each instance all legs sailed by all vessels employed.

BC The percentage of the total costs incurred which is attributable to bunker costs.

All the instances in the scenario analysis were solved to optimality. The only exceptions are the instances corresponding with 2- and 3-weeks time horizons. For these instances, the statistics used for the scenario analysis refer to the best integer solution obtained within 1 hour. Even if these instances were not solved to optimality, the gaps were all under 1% for the 2-weeks time horizon, and under 10% for the 3-weeks time horizon.

Table 4 shows the sensitivity of the solutions to variations in bunker price. As it can be expected, the share of costs attributable to bunker price is the most affected KPI (Key Performance Indicator). Revenue also benefit greatly from lower bunker costs, by exhibiting a wide gap between the lowest and highest values. It is interesting to notice that lower bunker prices allow for slightly faster speeds; this, in turn, could mean being able to serve more ports, thereby further increasing revenue and coverage. This can be seen for the \$250 and \$300 prices, where a +5.30% in the number of services means that one more service was performed.

Notice that the insertion of one port can alter the overall shape of the rotations considerably. Therefore, it can be difficult to add one additional service at a later time, after the network design has been decided, just because of a decrease in bunker prices which now makes that service profitable. For this reason, we advise network planners to first consider network layouts with more services, and to try increase the profitability of the routes by means of fuel hedging, contracting lower port fees, etc.

Table 5 analyses the impact of speed optimisation in the network plan. Notice how considering three rather than just one fixed speed has a great impact on all the KPIs considered. Not including a speed decision means obtaining solutions with −24% revenue and fewer served ports. Notice that the average speed

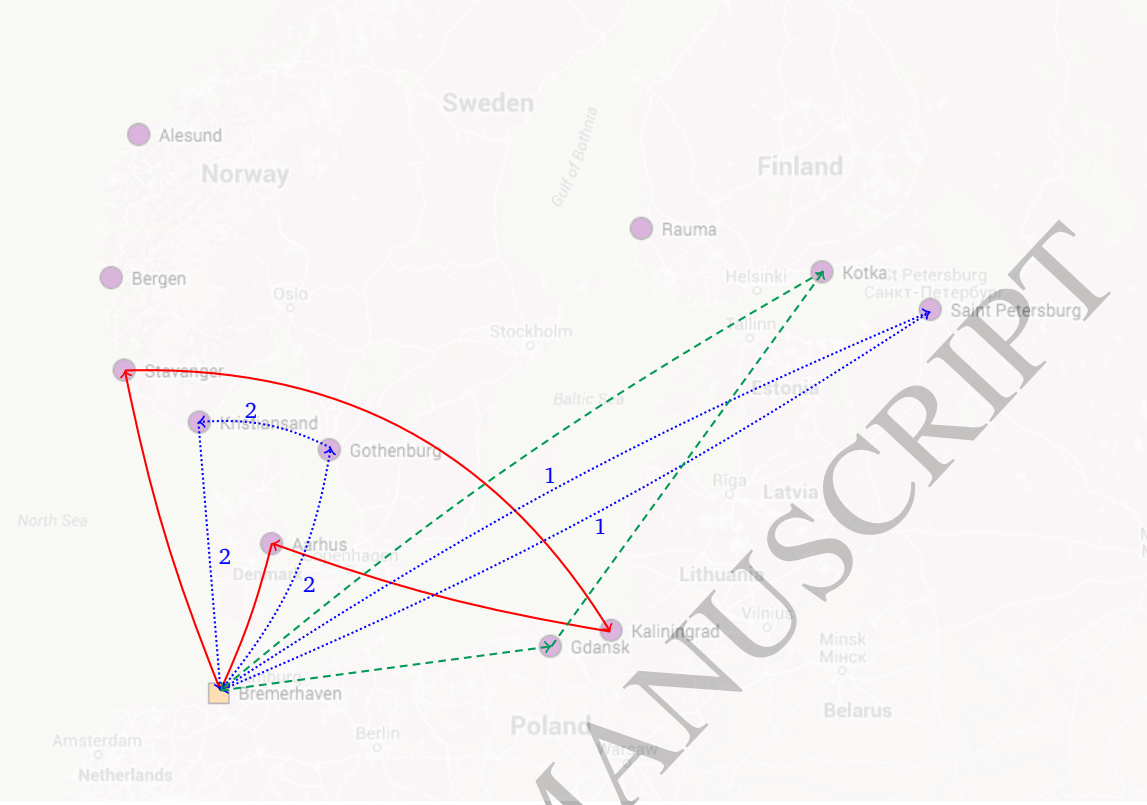


Figure 8: Optimal solution for instance Baltic1 with a 2-week time horizon.

is lower when only considering 1 speed (in this case the speed considered was the average between the maximum and minimum speeds provided by the LinerLib for each vessel) showing that the advantages of speed optimisation cannot be simply attributed to low-speed steaming, but rather to a smart combination of low-speed and high-speed legs which, on average, brings the speed up and allows a vessel to serve more ports in a single route. In some instances, using only one speed meant the deployment of one fewer vessel, as a route became either infeasible or unprofitable.

On the other hand, using five speed values has a positive effect on the revenue and the number of services performed. The difference is not as big as the one noted before (e.g. in terms of revenue generated) and the route shapes are very similar between the solutions with 3 and 5 speeds. Finally, the average speed is also very close between these two group of instances. In short, we see no downsides in using a higher number of speeds, as all instances could still be solved to optimality, but the planner should not expect to see big improvements just by increasing the granularity of speed discretisation beyond a certain point.

Other interesting observations can be made by considering Table 6, which shows the impact of demand. First of all notice that, as expected, less demand means less revenue for the operator. However, notice how revenue multiplier 0.8, corresponding to a reduction in demand of -20%, gives a reduction in revenue of just -8.52%. This is because, by using the same number of ships (column "SU") the optimal route increases coverage (column "Srv") as an effective countermeasure. A further reduction in demand, with multipliers 0.6 and 0.4 is, instead, excessive and cannot be counteracted as effectively.

The increased coverage seen for demand multiplier 0.8 is due to the fact that routes which were infeasible because of capacity restrictions, now become feasible. Compare, for example, the optimal solutions to instance Baltic1 with multipliers 1.0 (Figure 5) and 0.8 (Figure 7); in the second network plan, a vessel

serves the port of Kotka which was previously unserved. The ports of Gothenburg and Kotka are served by a vessel with capacity 800TEU; the sum of their pickup demands is 822TEU, but it reduces to just 658TEU when the multiplier is 0.8, thereby making the route feasible.

In summary, we can notice that a decrease in the demand corresponds to a roughly proportional decrease in the total revenue, but the planner can respond with better fleet utilisation and wider coverage. This shows that restructuring the routes can be an effective countermeasure during extended periods of low demand.

For the scenario with an increase in demand, on the other hand, we notice a modest increase in revenue (6.7% compared with the 20% higher demand) due to substantial cost reductions: fewer ports are now served, using fewer ships, and sailing at considerably lower speeds. This cost reduction, coupled with increased earnings due to the high demand, pushes the revenue up. It is interesting to see, however, how both a -40% reduction, and a $+20\%$ increase in demand can both lead to the same reduction in the size of the deployed fleet, albeit for different reasons.

In Table 7 we report the results of allowing to outsource some service. It can be noted that admitting this possibility only resulted in small changes in the generated routes. The major variation is recorded in the generated revenue, as ports that are not served earn nothing in the base case, while they earn a small fraction of their revenue in the other cases.

It is interesting to notice that, when keeping 10% of the revenue, the total number of services performed increased: the port of Saint Petersburg, which was served with a dedicated vessel, was instead outsourced and the vessel was used to increase coverage of other ports. This hints to the fact that the intuitive rule of thumb of focusing on high-demand (and, therefore, high-revenue) ports in the strategic phase, and delegating the decision of buying capacity on competitors to the tactical or operational stages, can lead to sub-optimal results.

Finally, Table 8 shows what happens when we allow longer routes. In the base case, the routes can last up to one week and there are 6 vessels available; in the 2-week case, we have 3 vessels; in the 3-week case, we only have 2 available vessels. It appears that the optimal route duration from the point of view of revenue should be of 2 weeks, and that further increasing the time horizon to three weeks gives worse solutions, even though still slightly better than the 1-week base case.

While in the 1-week scenario we were using on average 3 vessels (out of 6 available), in the 2-week case we produce 3 rotations (thereby deploying all 6 vessels), and in the 3-week case we produce 2 rotations (again deploying all 6 vessels). The number of services performed increases, as do the highest load efficiency and the cargo travel distance. At the same time, longer routes allow for lower speeds and, therefore, a lower share of costs attributable to the bunker. We can compare the optimal solutions of instance Baltic1 for a 1-week (Figure 5) and for a 2-week time horizon (Figure 8). Notice how the vessel that is serving Saint Petersburg can now be reused in the second part of the route, after being unloaded at the hub, and proceeds to serving Gothenburg and Kristiansand. At the same time, the other two vessels also perform longer routes.

This is probably the most impactful design decision, as the variation in earned revenue gets up to $+18.70\%$. Considering that the time horizon length is mostly an operator's decision which does not depend on external factors, this can surely be the most critical decision in the design of a feeder network.

7. Conclusions

In this paper we proposed an exact algorithm for the solution of the Feeder Network Design Problem. The algorithm can handle instances of realistic size and either solves them to optimality, or finds a solution close to the computed lower bound. The modelling framework is able to describe many real-world constraints and, as such, has been used to perform scenario analysis with the objective to derive general guidelines for network planners.

In particular, we assessed: (1) The impact of bunker price on the profitability of the services; we advise the planners to prioritise wider service coverage and fuel negotiations options. (2) The importance of leg-by-leg speed optimisation; to this end, while slow steaming is a consolidated practice for inter-continental services, we show that a combination of slower and faster sailing speeds is more apt for feeder networks. (3) The effect of demand fluctuations; we have showed that demand is a crucial factor in determining profitability, but the detrimental effects of a prolonged period of low demand can be reduced if the planner responds with a suitable network restructuring. (4) Outsourcing services by buying capacity from competitors can have deep effects on the network design; capacity availability on competitors, however, can be volatile and therefore we advise prudence when trying to incorporate this decision at the strategic level. (5) Designing longer rotations and deploying more vessels to each of them, can have a strong positive impact on profitability; however, the relationship is not linear, and the planner must perform an accurate analysis to determine the optimal rotation length.

As for future research avenues, we would like to retrieve realistic data for scenarios with more ports, to better asses which are the largest instances that the algorithm can solve to optimality. Furthermore, we would like to test the validity of the proposed approach for similar problems, such as the TOP with pickup and delivery for which, at the best of our knowledge, only one heuristic algorithm has been proposed [5].

Acknowledgments

We are grateful to the support from Optimization Manager Mikkel M. Sigurd from Mærsk Line, for fruitful discussions which made this work possible. Stefan Røpke was supported by The Danish Strategical Research Council and The Danish Energy Technology Development and Demonstration Program (EUDP) under the ENERPLAN and GREENSHIP project. We are also grateful to the anonymous referees for their comments, which greatly contributed to the improvement of the present work.

Appendix A. Detailed route description

In this section we describe in detail which ports have been visited and whether pickup or delivery was performed, relative to the routes represented in Figures 5 and 6. We list the sequence of ports, then a P and/or a D denoting whether pickup and delivery operations were performed, and then the ship capacity usage after visiting that port.

Routes in the Baltic scenario:

- Hub DEBRV, 366/450 → RUKGD, PD, 105/450 → PLGDK, PD, 238/450 → Hub DEBRV
- Hub DEBRV, 800/800 → RULED, PD, 800/800 → Hub DEBRV
- Hub DEBRV, 521/800 → DKAAR, PD, 462/800 → NOSVG, PD, 429/800 → Hub DEBRV, 597/800 → SEGOT, PD, 660/800 → Hub DEBRV

Routes in the Western African scenario:

- Hub ESALG, 565/800 → SNDKR, PD, 189/800 → Hub ESALG
- Hub ESALG, 409/450 → GNCKY, PD, 210/450 → LRMRW, PD, 111/450 → GHTKD, P, 338/450, Hub ESALG
- Hub ESALG, 412/450 → TGLFW, D, 0/450 → CIABJ, P, 326/800 → Hub ESALG
- Hub ESALG, 800/800 → NGAPP-1, PD, 800/800 → Hub ESALG
- Hub ESALG, 773/800 → NGAPP-2, D, 0/800 → TGLFW, P, 120/800 → Hub ESALG

- Hub ESALG, 683/800 → BJCOO, PD, 94/800 → Hub ESALG
- Hub ESALG, 402/450 → CMDLA, PD, 414/450 → GALBV, PD, 340/450 → Hub ESALG
- Hub ESALG, 756/800 → CGPNR, P, 775/800 → CDBOA, D, 715/800 → AOLAD, PD, 21/800 → CDMAT, P, 38/800 → Hub ESALG

References

[1] Richa Agarwal and Özlem Ergun. Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science*, 42(2):175–196, 2008.

[2] Martin Andersen. *Service Network Design and Management in Liner Container Shipping Applications*. PhD thesis, Danish Technical University, 2010.

[3] Claudia Archetti, Dominique Feillet, Alain Hertz, and Maria Grazia Speranza. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60(6):831–842, 2009.

[4] Claudia Archetti, M Grazia Speranza, and Daniele Vigo. Vehicle routing problems with profits. In Paolo Toth and Daniele Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2014.

[5] DG Baklagis, G Dikas, and I Minis. The team orienteering pick-up and delivery problem with time windows and its applications in fleet sizing. *RAIRO-Operations Research*, 50(3):503–517, 2016.

[6] Maria Battarra, Artur Pessoa, Anand Subramanian, and Eduardo Uchoa. Exact algorithms for the traveling salesman problem with draft limits. *European Journal of Operational Research*, 235(1):115–128, 2014.

[7] Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4OR: A Quarterly Journal of Operations Research*, 5(3):211–230, 2007.

[8] Berit Brouer, Fernando Alvarez, Christian Plum, David Pisinger, and Mikkel Sigurd. A base integer programming model and benchmark suite for liner-shipping network design. *Transportation Science*, 48(2):281–312, 2013.

[9] Pierre Cariou. Is slow steaming a sustainable means of reducing CO2 emissions from container shipping? *Transportation Research Part D: Transport and Environment*, 16(3):260–264, 2011.

[10] Ching-Chih Chang and Chih-Min Wang. Evaluating the effects of speed reduce for shipping costs and CO2 emission. *Transportation Research Part D: Transport and Environment*, 31:110–115, 2014.

[11] Marielle Christiansen, Kjetil Fagerholt, and David Ronen. Ship routing and scheduling: Status and perspectives. *Transportation science*, 38(1):1–18, 2004.

[12] Marielle Christiansen, Kjetil Fagerholt, Bjørn Nygreen, and David Ronen. Maritime transportation. *Transportation*, 14:189–284, 2006.

[13] Marielle Christiansen, Kjetil Fagerholt, Bjørn Nygreen, and David Ronen. Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483, 2013.

[14] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164, 1981.

[15] Mauro Dell'Amico, Giovanni Righini, and Matteo Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2006.

[16] Guy Desaulniers, François Lessard, and Ahmed Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008.

[17] Maritime Research Drewry. Seaborne trade annual report 2013. Technical report, Drewry, 2014.

[18] Yvan Dumas, Jacques Desrosiers, and Francois Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22, 1991.

[19] Jørgen Glomvik Rakke, Marielle Christiansen, Kjetil Fagerholt, and Gilbert Laporte. The traveling salesman problem with draft limits. *Computers & Operations Research*, 39(9):2161–2167, 2012.

[20] Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In *Column generation*, chapter 2, pages 33–65. Springer, 2005.

[21] Christos Kontovas. The green ship routing and scheduling problem (gsrsp): A conceptual approach. *Transportation Research Part D: Transport and Environment*, 31:61–69, 2014.

[22] Marine Intelligence Unit Lloyd's. Measuring global seaborne trade. Technical report, Lloyd's, 2009.

[23] Berit Løfstedt, Fernando Alvarez, Christian Plum, David Pisinger, and Mikkel Sigurd. An integer programming model and benchmark suite for liner shipping network design. Technical Report 19, DTU, Technical University of Denmark, 2010.

[24] Enrico Malaguti, Silvano Martello, and Alberto Santini. The traveling salesman problem with pickups, deliveries, and draft limits. *Omega*, In press, 2017.

[25] Qiang Meng, Shuaian Wang, Henrik Andersson, and Kristian Thun. Containership routing and scheduling in liner shipping: overview and future research directions. *Transportation Science*, 48(2):265–280, 2013.

[26] Judith Mulder and Rommert Dekker. Methods for strategic liner shipping network design. *European Journal of Operational Research*, 235(2):367–377, 2014.

[27] Christian Plum, David Pisinger, Juan-José Salazar-González, and Mikkel Sigurd. Single liner shipping service design. *Computers & Operations Research*, 45:1–6, 2014.

[28] Christian Plum, David Pisinger, and Mikkel Sigurd. A service flow model for the liner shipping network design problem. *European Journal of Operational Research*, 235(2):378–386, 2014.

[29] Harilaos Psaraftis and Christos Kontovas. Speed models for energy-efficient maritime transportation: A taxonomy and survey. *Transportation Research Part C: Emerging Technologies*, 26:331–351, 2013.

[30] Harilaos Psaraftis and Christos Kontovas. Ship speed optimization: Concepts, models and combined speed-routing scenarios. *Transportation Research Part C: Emerging Technologies*, 44:52–69, 2014.

[31] Line Blander Reinhardt and David Pisinger. A branch and cut algorithm for the container shipping network design problem. *Flexible Services and Manufacturing Journal*, 24(3):349–374, 2012.

[32] Alberto Santini. maritime-vrp v1.3, Jun 2017. URL https://doi.org/10.5281/zenodo.802330.

[33] Martin Savelsbergh and Marc Sol. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490, 1998.

[34] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. Siam, 2014.

[35] Shuaian Wang and Qiang Meng. Sailing speed optimization for container ships in a liner shipping network. *Transportation Research Part E: Logistics and Transportation Review*, 48(3):701–714, 2012.

[36] Shuaian Wang and Qiang Meng. Liner shipping network design with deadlines. *Computers & Operations Research*, 41:140–149, 2014.

[37] The World Shipping Council. The liner shipping industry and carbon emission policies. Technical report, The World Shipping Council, 2009.