SN Operations Research Forum (2020) 1: 28
https://doi.org/10.1007/s43069-020-00027-y

ORIGINAL RESEARCH

A Branch-and-Price Algorithm for the Liner Shipping Network Design Problem

Kristian Thun[1] · **Henrik Andersson**[1] · **Magnus Stålhane**[1]

Received: 20 March 2020 / Accepted: 16 September 2020 / Published online: 21 October 2020

Abstract

Maritime transportation is the backbone of the global economy and one of its most important segments is liner shipping. To design a liner shipping network is notoriously difficult but also very important since an efficient network can be the difference between prosperity and bankruptcy. In this paper, we propose a branch-and-price algorithm for the liner shipping network design problem, which is the problem of designing a set of cyclic services and to deploy a specific class of vessels to each service so that all demand can flow through the network at minimal cost. The proposed model can create services with a complex structure and correctly calculate the transshipment cost. The formulation of the master problem strengthens a known formulation with valid inequalities. Because of multiple dependencies between ports that are not necessarily adjacent and no defining state at any of the ports, the subproblem is formulated and solved as a mixed integer linear program. Strategies to improve the solution time of the subproblem are proposed. The computational study shows that the algorithm provides significantly tighter lower bounds in the root node than existing methods on a set of small instances.

Keywords Liner shipping · Network design · Branch-and-price

1 Introduction

Ever since man first set sails across the oceans, transporting goods between continents has been an important part of the economy. Today, maritime transportation is

This article belongs to the Topical Collection on *Decomposition at 70*

✉ Henrik Andersson
henrik.andersson@ntnu.no

[1] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Gløshaugen, Alfred Getz vei 3, 7491 Trondheim, Norway

the backbone of the global economy and United Nations Conference on Trade And Development (UNCTAD) has in its review of maritime transportation shown that the volumes, counted in tonnes loaded, of international maritime trade has almost doubled since 2000, from slightly above $6 \cdot 10^9$ tonnes loaded in 2000 to almost $12 \cdot 10^9$ in 2018 [18]. Many types of goods are transported and many different vessels are used to do it, but maritime transportation is traditionally divided into three segments: industrial shipping, tramp shipping, and liner shipping, [9]. An industrial operator controls its own fleet of vessels and strives to minimize the costs of transporting its own cargoes. An operator within tramp shipping has similarities with taxi services, as the vessels follow the cargoes that become available in the market. Liner shipping resembles bus line operations since the vessels follow published schedules and itineraries. UNCTAD among others divided the goods transported into four categories; tanker trade, main bulk, other dry cargo, and containers [18]. The growth of global containerized trade has been around 5.8% yearly since 2000 and has as of 2018 reached 152 million 20-ft equivalent units compared with 63 millions in 2000. Most of the containerized trade is within liner shipping and the economic importance of liner shipping is therefore obvious.

Many different planning problems within liner shipping have been analyzed using operations research. Meng et al. [11] outline some of these, including fleet size and mix and network design on the strategic level, fleet deployment and speed optimization on the tactical level, and cargo routing and rescheduling on the operational level. For more comprehensive overviews of planning problems within liner shipping, the reader is referred to [5], [17], [10], and [6]. A liner shipping network consists of a number of cyclic routes, called services, visiting two or more ports each. Each service is assigned a given vessel class and sailed with a given frequency. Demand is given between pairs of ports, and can be transshipped at one or more intermediate ports. This means that the vessel picking up the demand may not be the vessel delivering it, but instead that it can be handled by any service, even if neither origin nor destination ports are part of that service. Designing a liner shipping network is a notoriously hard problem called the liner shipping network design problem (LSNDP); see [4] for an introduction to LSNDP. The practical aspects that can be included when defining the LSNDP are numerous and there are therefore many versions of the problem. Common aspects are, according to [6], transit time constraints, transshipment costs, rejected demands, speed optimization, and the structure of the services. The methods used to solve the LSNDP include both heuristics and exact methods, but most of them are based on a mathematical formulation of the problem and mathematical programming techniques. Here, we review the exact methods that have been proposed; for a detailed description of solution methods for the LSNDP, both heuristics and exact methods, the reader is referred to [6].

The seminal paper by [1] is the first that presents an exact solution method for the LSNDP. The problem is formulated over a time-space network, where a weekly frequency is assumed. Transit time is not included, and transshipment costs are calculated once the network has been designed and are not included in the design phase.

Demand can be rejected, and there is a revenue for fulfilled demand. Complex services are possible as long as the visits to one port are not on the same day. In [13], the authors notice that disregarding transshipment costs is a weakness and propose a four-index formulation to remedy this. Butterfly services, i.e., services where one port can be visited twice, are handled by enumerating the order of the arcs of a service. Transit time is not included, and each service is served by a single vessel. All demand must be served. In [12], the authors notice that butterfly services may be too restrictive and propose a formulation allowing for multiple visits to multiple ports on each service. Weekly frequencies on the services are assumed and demand can be rejected. An analysis of the effect of different structures of the services is conducted by [16]. Simple services, i.e., no ports are allowed to be visited more than once on each service, are compared with butterfly services and even more complex services. The complex structure is modeled using a two-layer network, where each port is represented in both layers and the connection between the layers is only between nodes corresponding to the same port. This allows for many different structures such as butterfly and pendulum where all ports except two are visited twice; see Fig. 1. Weekly frequencies are assumed and all demand must be served. In a feeder network, all demand either originates from or is destined for a single port. The design problem on these networks is studied by [14]. Transshipment is not allowed and each demand is either accepted or rejected. In [2], the authors present a new compact formulation for the same version of the problem as studied by [16]. New valid inequalities are introduced to strengthen the formulation. One of their main findings is that the gain from using a two-layer formulation compared with a one-layer formulation is on average 2–4% at the cost of a significant increase in solution time.

This paper presents an improved formulation and a new branch-and-price algorithm for the LSNDP. The work in [16] is extended with new valid inequalities and a new solution method for the subproblem is developed. The rest of the paper is organized as follows. In Section 2, a problem description is given and the master problem is formulated together with valid inequalities. Section 3 outlines the proposed branch-and-price algorithm and Section 4 provides a computational study. Finally, Section 5 gives some concluding remarks.

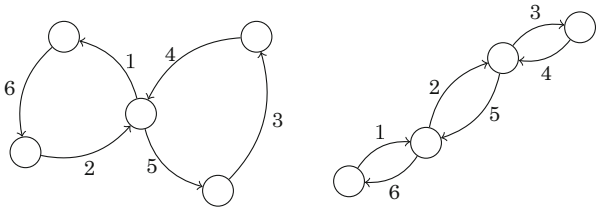

Fig. 1 Two examples of complex service structures. Left: A butterfly service where one port is visited twice. Right: A pendulum service where all ports except the ends are visited twice. The numbers by the arcs show the order of traversal

2 Problem Description and Mathematical Formulation

Following the classification of liner shipping network design problems in [6], the problem studied here does not contain transit time constraints, speed optimization, or the opportunity to reject demand. It includes transshipment costs and the possibility to use complex route structures. We now give a description of the problem, the notation, and the mathematical formulation.

The liner shipping network design problem studied in this paper consists of a set of ports $\mathcal{P}$. There are weekly demands, D_{ij}, to be transported from port i to port j and a given fleet of vessels to perform the transportation. All demands must be fulfilled; hence, it is not possible to reject demand. The vessels are divided into vessel classes, $\mathcal{V}$, and there is an upper bound on the number of vessels, Y_v, available from each class v. The vessels are assigned to services, $\mathcal{S}$, where each service represents a given sequence of ports. The duration of a service is bounded, but the services can be complex. For any combination of vessel class and service, we let A_{vs} be the number of vessels needed to keep a given (typically weekly) frequency on that service.

Since it is possible to split a demand between different services and also to transship, there are multiple ways to load and unload goods along a service. There is no bound on the transit time for the demands, and they can be transshipped as many times as needed. We associate a set of delivery patterns $\mathcal{W}_{vs}$ with each possible combination of service s and vessel class v. For a given delivery pattern $w \in \mathcal{W}_{vs}$, let Q^L_{ijvsw} be the number of units with final destination j that are loaded onto service s of vessel class v at port i per week. This includes units that are loaded at their port of origin, as well as units that have been transshipped from another service. Similarly, we define Q^U_{ijvsw} as the number of units with final destination j that are unloaded from service s of vessel class v and delivery pattern w at port i per week. To minimize the number of delivery patterns in $\mathcal{W}_{vs}$, we may use the property that every convex combination of feasible delivery patterns is a feasible delivery pattern; and thus, we only need to explicitly represent the extreme delivery patterns in order to express all of them. In [7] and [15], the authors introduce deliver patterns in a similar way in problems with split deliveries. For each combination of service s, vessel class v, and delivery pattern w, we denote C_{vsw} as the total cost of using service s of vessel class v and delivery pattern w. The cost includes a fixed cost for operating the vessel, sailing costs, port fees, and costs for loading and unloading goods.

The problem then consists of selecting a set of services, and assign a vessel class and delivery pattern to each selected service, such that all demands are transported from their origin port to their destination port. To model this, we introduce variables p_{vs} which is 1 if service s of vessel class v is used and 0 otherwise, and y_{vsw} representing the number of times service s of vessel class v with delivery pattern w is used. Using these variables, and the notation defined above, a mathematical model of the problem may be defined as follows:

$$\min z = \sum_{v \in \mathcal{V}} \sum_{s \in \mathcal{S}} \sum_{w \in \mathcal{W}_{vs}} C_{vsw} y_{vsw} \tag{1}$$

Subject to:

$$\sum_{v\in\mathcal{V}}\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}}(Q^L_{ijvsw}-Q^U_{ijvsw})y_{vsw} = D_{ij} \qquad i,j\in\mathcal{P}, i\neq j \tag{2}$$

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}} A_{vs}y_{vsw} \leq Y_v \qquad v\in\mathcal{V} \tag{3}$$

$$\sum_{w\in\mathcal{W}_{vs}} y_{vsw} - p_{vs} = 0 \qquad v\in\mathcal{V}, s\in\mathcal{S} \tag{4}$$

$$p_{vs} \in \{0,1\} \qquad v\in\mathcal{V}, s\in\mathcal{S} \tag{5}$$

$$y_{vsw} \geq 0 \qquad v\in\mathcal{V}, s\in\mathcal{S}, w\in\mathcal{W}_{vs} \tag{6}$$

The objective function (1) minimizes the total cost associated with the services used. The flow balance constraints (2) make sure that the flow of goods is preserved. Since the destination of the goods unloaded when using a service is known, transshipment costs can be handled in the generation of the service and delivery pattern and can therefore be omitted from the master problem. Constraints (3) ensure that there are enough vessels of each class for the services used. The convexity constraints (4) state that different delivery patterns for the same service can be weighted together, and constraints (5) are binary restrictions. The non-negativity of the variables is expressed in constraints (6).

To illustrate the formulation and the variables, assume that we have a small network of six ports; see Fig. 2. Three services are shown; the dashed service, s_1, is operated by a vessel with capacity 12 from class v_1 while the solid service s_2 and the dotted service s_3 are both operated by a vessel with capacity 7 from class v_2. There is one demand defined, $D_{15} = 10$, i.e., 10 units are requested from port 1 to port 5. The solution we illustrate is to transport 10 unit from port 1 using service s_1; 7 units are transshipped at port 3 and continue to the destination at port 5 using service s_2. The last 3 units are transshipped at port 4 and continue to the destination using service s_3. Only allowing extreme delivery patterns, i.e., patterns that cannot be expressed as convex combinations of other patterns, we define two patterns w_1 and w_2 for service s_1. The non-zero coefficients of these patterns are $Q^L_{15v_1s_1w_1} = 10$, $Q^U_{35v_1s_1w_1} = 10$ and $Q^L_{15v_1s_1w_2} = 10$, $Q^U_{45v_1s_1w_2} = 10$. Correspondingly, we get $Q^L_{35v_2s_2w_3} = 7$ for

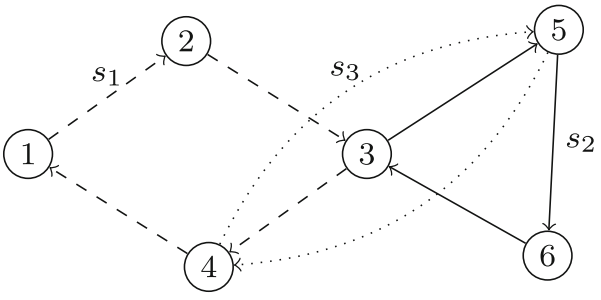

Fig. 2 A small network with six ports operated by three service

the pattern w_3 for service s_2. Note that the coefficient for unloading at the destination is excluded. Since the extreme delivery pattern w_4 for service s_3 with coefficient $Q^L_{45v_2s_3w_4} = 7$ handles too many units, we also need to define a pattern w_5 for service s_3 with no non-zero coefficients. Using the defined variables, the solution is now expressed in Table 1.

2.1 Strengthening the Formulation

In [2], the authors successfully applied the method of expressing the convex hull of a low-dimensional polyhedron as a convex combination of its extreme points to strengthen their formulation of the LSNDP. They specifically targeted the feasible region defined by capacity constraints over subsets of ports and called this an inner representation of the polyhedron. We have adopted this idea and adjusted it for the master problem. The set of subsets of $\mathcal{P}$ is denoted $\mathcal{U}$, and let $\mathcal{P}^S_u$ be the subset of ports under consideration and $\overline{\mathcal{P}}^S_u$ the complement of $\mathcal{P}^S_u$. The least quantity of goods that must be transported either in or out of $\mathcal{P}^S_u$ is $\max\{\sum_{i\in\mathcal{P}^S_u}\sum_{j\in\overline{\mathcal{P}}^S_u} D_{ij}, \sum_{i\in\overline{\mathcal{P}}^S_u}\sum_{j\in\mathcal{P}^S_u} D_{ij}\}$.

If the number of vessel classes, $|\mathcal{V}|$, is small, we can generate all points $\chi^S_u = (\chi^S_{u1}, \ldots, \chi^S_{u|\mathcal{V}|})$ where χ^S_{uv} is the number of times a vessel of class v enters $\mathcal{P}^S_u$ that fulfill:

$$\sum_{v\in\mathcal{V}} Q_v \chi^S_{uv} \geq \max\left\{\sum_{i\in\mathcal{P}^S_u}\sum_{j\in\overline{\mathcal{P}}^S_u} D_{ij}, \sum_{i\in\overline{\mathcal{P}}^S_u}\sum_{j\in\mathcal{P}^S_u} D_{ij}\right\} \tag{7}$$

and is part of the convex hull of the feasible region defined by constraint (7) intersected with $\mathcal{X} = \{\chi^S_{uv} \in \mathbb{N}\cup\{0\}, v \in \mathcal{V}\}$. Define Λ^S_u to be the number of such points for subset $\mathcal{P}^S_u$ and let χ^S_{luv} be the value χ^S_{uv} for point l. Since the sequence of ports visited is known for each service, we can define G^S_{us} to be the number of times service s enters subset $\mathcal{P}^S_u$. With $\mathcal{P}^S_u = \{1, 3, 5\}$ and service $s = 1 \to 2 \to 4 \to 5 \to 3 \to 6 \to 1$ we get $G^S_{us} = 2$ since s enters $\mathcal{P}^S_u$ twice, $6 \to 1$ and $4 \to 5$. With this, the constraints presented in [2] can be adjusted to:

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}} G^S_{us} y_{vsw} \geq \sum_{l=1}^{\Lambda^S_u} \chi^S_{luv}\lambda^S_{lu} \qquad v \in \mathcal{V}, u \in \mathcal{U} \tag{8}$$

$$\sum_{l=1}^{\Lambda^S_u} \lambda^S_{lu} \geq 1 \qquad u \in \mathcal{U} \tag{9}$$

$$\lambda^S_{lu} \geq 0 \qquad u \in \mathcal{U}, l = 1, \ldots, \Lambda^S_u \tag{10}$$

Table 1 A possible solution for $D_{15} = 10$ in Fig. 2

v_1, s_1	$p_{v_1s_1} = 1$	$y_{v_1s_1w_1} = 0.7$, $y_{v_1s_1w_2} = 0.3$
v_2, s_2	$p_{v_2s_2} = 1$	$y_{v_2s_2w_3} = 1$
v_2, s_3	$p_{v_2s_3} = 1$	$y_{v_2s_3w_4} = 3/7$, $y_{v_2s_3w_5} = 4/7$

where λ_{lu}^{S} is the fraction of point l of subset u. Constraints (8)–(10) can be added to the master problem (1)–(6) to strengthen it.

3 Solution Method

A major challenge with the mathematical model presented in Section 2 is that it contains one y-variable, henceforth referred to as a column, for each combination of vessel class, service, and delivery pattern. The total number of columns thus grows very fast with the number of ports and demands and it becomes cumbersome to explicitly generate all columns, even for problems with just a few ports and demands. To circumvent this problem, we have developed a solution method based on branch-and-price [3] to effectively solve this problem. Branch-and-price is a branch-and-bound (B&B) tree search algorithm, where the linear program (LP) solved to obtain dual bounds for each node of the B&B-tree is solved using column generation [8].

Column generation is a method to solve linear programs that starts by solving a restricted version of the problem, containing only a subset of the columns (variables). This problem is usually referred to as the restricted master problem (RMP), and solving the RMP gives a pessimistic bound on the optimal value of the LP. To improve this pessimistic bound, one or more subproblems are solved, which finds the column with the lowest reduced cost (assuming that the LP is a minimization problem) based on the dual information from the solution of the RMP. If one or more negative reduced cost columns are found, they are added to the RMP, which is then re-solved to obtain an improved pessimistic bound and a new dual solution, and the previous step is repeated. If the subproblems cannot obtain any negative reduced cost columns, then we have proved that the current solution of the RMP is in fact optimal for the LP relaxation of the full problem in the current node, and we have obtained a valid dual bound for that node. The solution is checked for integrality, and if it is not integral, new nodes are created using an appropriate branching strategy.

In the following, we describe first how to price (calculate the reduced cost of) a column in Section 3.1, before giving a mathematical formulation of the subproblems that are solved to find the minimum reduced cost columns in Section 3.2. Then, in Section 3.3, we describe two heuristic strategies to find negative reduced cost columns quickly, before discussing the branching strategies, and their impact on the subproblem in Section 3.4.

3.1 Pricing of a Column

To define the reduced cost of a column, we introduce β_{ij}^{*} as the dual value from the flow conservation constraints (2), γ_{v}^{*} as the dual value from constraints (3), and α_{uv}^{*} as the dual value from constraints (8). The dual values from the other constraints do not affect the reduced cost of a column and are therefore omitted here. There are port fees related to vessels visiting the ports and handling costs related to loading and unloading at the ports. The cost structure for the vessels is a fixed cost for each

vessel used and distance-dependent sailing costs. We denote the costs in the following way; C_v^F is the weekly fixed cost for operating a vessel of class v, C_i^K is the cost of loading/unloading one unit of goods at port i, and C_{ijv} is the sailing cost between ports i and j including the port fee at port j for vessel class v. Service s is represented as a cyclic list of port visits $\pi_1 \rightarrow \pi_2 \rightarrow \cdots \rightarrow \pi_{\rho_s}$, where ρ_s is the number of port visits of service s. To simplify the notation required to describe the last leg of a cycle back to the starting point, we define $\pi_{\rho_s+1} = \pi_1$. The reduced cost of a column representing service s with delivery pattern w for vessel class v can now be stated:

$$\overline{C}_{vsw} = (C_v^F + \gamma_v^*)A_{vs} + \sum_{i=1}^{\rho_s} C_{\pi_i \pi_{i+1} v} + \sum_{u \in \mathcal{U}} \alpha_{uv}^* G_{us} + \\ + \sum_{i=1}^{\rho_s} \sum_{j \in \mathcal{P}} \left((C_{\pi_i}^K + \beta_{\pi_i j}^*) Q_{\pi_i j v s w}^L + (C_{\pi_i}^K - \beta_{\pi_i j}^*) Q_{\pi_i j v s w}^U \right) \quad (11)$$

3.2 Subproblem

The subproblems used to generate columns for the RMP can be stated for each vessel class. Each subproblem is formulated over a two-layer graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where each physical port is represented by exactly one node in each layer. Each layer is a complete subgraph and the layers are only connected by arcs between the nodes representing the same port. Figure 3 shows the ports (the left part) and the corresponding graph (the middle part). Note that all arcs between nodes in the same layer are omitted to increase readability. All simple cycles in the graph are defined as feasible services. With two layers, each port can be visited at most twice on a service, but not all services fulfilling this criterion are feasible. The right part of Fig. 3 shows an example of a service that visits no port more than twice that cannot be represented as a simple cycle in the graph.

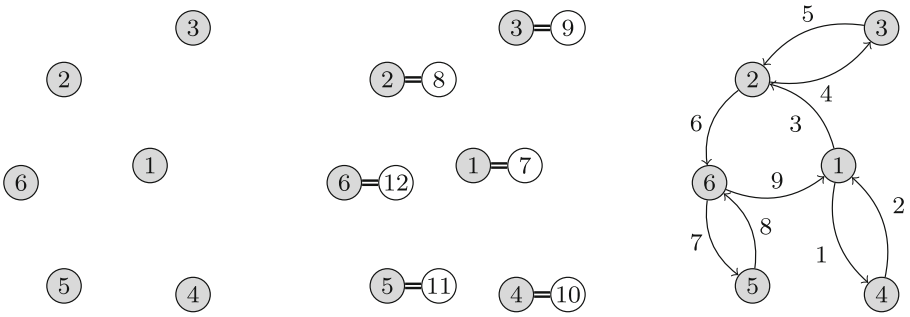

Fig. 3 The left part shows the location of the ports. The middle figure shows the two-layer graph. Each port is duplicated and the graph consists of two layers (slightly shifted and marked gray and white respectively). The double lines between the nodes representing the same port are the only connections between the layers. Each layer is a complete graph, but these arcs are omitted to increase readability. The right part is an example of a route that visits no port more than twice, but that cannot be represented as a simple cycle in the graph in the middle. The numbers by the arcs is the order of arc traversal

The graph has two layers and we define the nodes of the graph as $\mathcal{N} = \{1, \ldots, 2|\mathcal{P}|\}$. The nodes are divided into an upper layer consisting of nodes $1, \ldots, |\mathcal{P}|$ and a lower layer consisting of nodes $|\mathcal{P}| + 1, \ldots, 2|\mathcal{P}|$. Each port i is associated with both node i and $|\mathcal{P}| + i$. The set of arcs $\mathcal{A}$ is the union of all arcs connecting nodes in the upper layer, all arcs connecting nodes in the lower layer, and arcs connecting the layers, i.e., $(i, |\mathcal{P}| + i)$ and $(|\mathcal{P}| + i, i)$ for all ports i. Hence, it is only possible to change layers by using arcs connecting nodes representing the same port. We also define $\mathcal{A}_u^S$ to be all arcs (i, j) for which $i \in \mathcal{P}_u^S$ and $j \in \overline{\mathcal{P}}_u^S$, i.e., all arcs leaving the subset $\mathcal{P}_u^S$ of ports. We introduce Q_v for the capacity of a vessel of class v and T_{ijv} for the sailing time in weeks between port i and j for a vessel of class v. The maximum length of a service, in weeks, is restricted by $\overline{T}$.

Both the construction of the service and the amount to load and unload in the ports are decisions in the subproblem. We therefore introduce x_{ij} as a binary variable stating if arc (i, j) is part of the service or not, q_{ik}^L as the amount of goods with final destination k that is loaded at node i and q_{ik}^U as the amount of goods with final destination k that is unloaded at node i. We also need the variable a for the number of vessels deployed on the service. To handle the capacity of the vessels, we define f_{ijk} for the amount of goods traveling along arc (i, j) with final destination k. Subtours, i.e., two or more disjoint cycles, are elusive in this formulation. We eliminate subtours by letting one of the visited nodes in the service be an imaginary depot. The flow of an imaginary product along the service is forced to increase for each port visit, and can only decrease at the depot. A subtour without the depot can therefore not exist. To handle this in the formulation, we introduce the binary variable w_i stating if node i is the imaginary depot of the service or not, and z_{ij} as the flow of an imaginary product from node i to j. Since each port is represented by two different nodes, we use $p(i)$ to denote the port associated with node i. The subproblem of the LSNDP for vessel class v can now be formulated:

$$\min z_{SP}(v) = (C_v^F + \gamma_v^*)a + \sum_{(i,j)\in\mathcal{A}} C_{ijv}x_{ij} + \sum_{u\in\mathcal{U}}\sum_{(i,j)\in\mathcal{A}_u^S} \alpha_{uv}x_{ij} + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{P}}(C_{p(i)}^K + \beta_{p(i)k}^*)q_{ik}^L + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{P}}(C_{p(i)}^K - \beta_{p(i)k}^*)q_{ik}^U \quad (12)$$

Subject to:

$$\sum_{(j,i)\in\mathcal{A}} x_{ji} - \sum_{(i,j)\in\mathcal{A}} x_{ij} = 0 \qquad i \in \mathcal{N} \quad (13)$$

$$\sum_{(j,i)\in\mathcal{A}} x_{ji} \leq 1 \qquad i \in \mathcal{N} \quad (14)$$

$$\sum_{i\in\mathcal{N}} w_i = 1 \quad (15)$$

$$z_{ij} - (|\mathcal{N}| - 1)x_{ij} \leq 0 \qquad (i, j) \in \mathcal{A} \quad (16)$$

$$\sum_{(i,j)\in\mathcal{A}} z_{ij} - \sum_{(j,i)\in\mathcal{A}} z_{ji} + |\mathcal{N}|w_i \geq \sum_{(i,j)\in\mathcal{A}} x_{ij} \qquad i \in \mathcal{N} \quad (17)$$

$$\sum_{(i,j)\in\mathcal{A}} f_{ijk} - \sum_{(j,i)\in\mathcal{A}} f_{jik} + q^L_{jk} - q^U_{jk} = 0 \qquad j \in \mathcal{N}, k \in \mathcal{P} \tag{18}$$

$$\sum_{k\in\mathcal{P}} f_{ijk} - Q_v x_{ij} \leq 0 \qquad (i,j) \in \mathcal{A} \tag{19}$$

$$f_{ijk} - H_{vk} x_{ij} \leq 0 \qquad (i,j) \in \mathcal{A}, k \in \mathcal{P} \tag{20}$$

$$\sum_{(i,j)\in\mathcal{A}} T_{p(i)p(j)v} x_{ij} \leq a \tag{21}$$

$$a \in \{0, 1, \ldots, \overline{T}\} \tag{22}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in \mathcal{A} \tag{23}$$

$$0 \leq f_{ijk} \leq \sum_{l\in\mathcal{N}} D_{lk} \qquad (i,j) \in \mathcal{A}, k \in \mathcal{P} \tag{24}$$

$$q^L_{ik}, q^U_{ik} \geq 0 \qquad i \in \mathcal{N}, k \in \mathcal{P} \tag{25}$$

The objective function (12) is the reduced cost of the column that is created. Constraints (13) are balance constraints and force all services to be closed loops while constraints (14) ensure that each node is visited at most once. Constraints (15)–(17) eliminate subtours. Constraint (15) states that exactly one node is the imaginary depot. Constraints (16) connect the flow on the imaginary product with the route and constraints (17) handle the flow balance, if node i is not the depot, the outgoing flow is one larger than the incoming, while the constraints are redundant if node i is the depot. Constraints (18) balance the flow of cargo entering and leaving a node, and capacity restrictions are enforce by constraints (19) and (20). We have here introduced $H_{vk} = \min\left\{Q_v, \sum_{l\in\mathcal{N}} D_{lk}\right\}$ to increase readability. That the number of vessels employed on the service must be enough to guarantee a weekly frequency is stated in constraints (21). Constraints (22) declare the number of deployed vessel to be integer and set the upper bound on the duration of the services. The other variables are declared in constraints (23)–(25).

If the optimal objective value to a subproblem is negative, the solution is used to create a new column that is added to the RMP. The cost of the new column is calculated as:

$$C_{vsw} = C^F_v a^* + \sum_{(i,j)\in\mathcal{A}} C_{ijv} x^*_{ij} + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{P}} C^K_{p(i)}(q^{L*}_{ik} + q^{U*}_{ik}) \tag{26}$$

The route defined by x^*_{ij} is transformed to a cyclic list of port visits, and a check is made to see if the service already exists. Since the cost of a column consists of both sailing costs and costs for loading and unloading, an already existing service with a new delivery pattern can be optimal in the subproblem. In this case, the index of the existing service is used and a new delivery pattern is associated with this service. The loading and unloading parameters Q^L_{ikvsw} and Q^U_{ikvsw} are directly taken from q^{L*}_{ik} and q^{U*}_{ik}, respectively, and the number of vessels deployed on the service, A_{vs} is assigned a^*. Finally, the parameter G_{us} is given the value $\sum_{(i,j)\in\mathcal{A}^S_u} x^*_{ij}$.

Henceforth, the subproblem defined by objective function (12) and constraints (13)–(25) is called the full subproblem. To increase the number of columns created with each time the full subproblem is solved, we supply the MIP solver with a callback routine to store all encountered solutions during the search which have a negative reduced cost. A new column is created for each of these solutions, so that a single iteration of solving the full subproblem may yield multiple new columns in the master problem.

3.3 Acceleration Strategies

A major success criterion for applying branch-and-price is that the structure of the subproblem(s) can be exploited. In most applications, the subproblems are solved using some form of dynamic programming, e.g., by using labeling algorithms. However, constructing cyclic sequences of nodes and assigning vessels and cargo flows to these sequences (which is the natural subproblem for the LSNDP) is not well suited for dynamic programming, as there are multiple dependencies between ports that are not necessarily adjacent in the sequence and no defining state at any of the nodes. This makes it difficult to design an algorithm adhering to Bellmann's optimality principle.

As it is not easy to design an effective exact solution method to solve the subproblems, it is important to design good heuristics to find most of the negative reduced cost columns quickly, thus minimizing the number of times we have to solve the subproblem as a MIP, which is known to be time consuming for large instances of the problem. We have therefore implemented the following two heuristics to obtain negative reduced cost columns from a given service:

- *Improve delivery patterns* There is (possibly) a large number of feasible delivery patterns associated with every service. The integral variables in the subproblem decide the port sequence (x_{ij}) and the number of vessels needed (a), i.e., the design of the service. Therefore, if we focus on a specific service, we only need to solve a linear program to find the best delivery pattern for that service. This can be done for each service individually.
- *Local search* We do a local search on an already existing service by either removing or adding a port visit. If port p is removed in the sequence $i \rightarrow j \rightarrow p \rightarrow k$, it becomes $i \rightarrow j \rightarrow k$. If j and k represent the same port, one of them is removed as well. If port p is added to the sequence $i \rightarrow j \rightarrow k$ after j, it becomes $i \rightarrow j \rightarrow p \rightarrow k$. This is not done if j or k represents the same port as p. After a port is added to or removed from a service, it is verified that the service represents a simple cycle in the two-layer graph. If the check is affirmative, a linear program is solved to find the best deliver pattern and the corresponding reduced cost.

Even though these heuristics are computationally fast, preliminary testing showed that it was too time consuming to run these heuristics for all services that are a part of at least one column in the RMP. Instead, we partition $\mathcal{S}$ into three disjoint subsets; $\mathcal{S}^{B+}$ is the set of all services that are part of a column with a positive value in the current basis, $\mathcal{S}^{B0}$ is the set of all services associated with a column that has a reduced

cost of zero and also a value of zero in the current basis, and $\mathcal{S}^N$ is the set of all services that are not part of any column in the basis.

The heuristics are run in a hierarchical fashion, and if at least one negative reduced cost column is found in one step, the algorithm returns it to the RMP, which is then re-solved. The hierarchy of these heuristics is as follows:

1. Improve the delivery patterns for all services in $\mathcal{S}^{B+}$
2. Improve the delivery patterns for all services in $\mathcal{S}^{B0}$
3. Perform local search on all services in $\mathcal{S}^{B+}$
4. Perform local search on all services in $\mathcal{S}^{B0}$

If none of these four steps returns a negative reduced cost column, we solve the mathematical model presented in Section 3.2 as a MIP using a commercial solver. However, since we only need a negative reduced cost column, we set a time limit on the MIP, and abort the solver if at least one negative reduced cost column has been found by that time, or as soon as one is found after this limit. Furthermore, we solve each MIP subproblem sequentially, and return to the RMP as soon as one of the MIPs has found a negative reduced cost column.

3.4 Branching

When a node has been solved to optimality, i.e., when no column with a negative reduced cost is found, it is checked with respect to the integrality restrictions (4). If the check is affirmative, we have a feasible integer solution; otherwise, we have to branch. Designing a branching scheme is intricate since it affects the structure of the subproblem. On one hand, the scheme should reach integer solutions fast, while on the other the changes to the subproblem should be small. Since the subproblems here are solved using a MIP solver, we allow for the structure to change by introducing branching constraints. We have adopted the branching scheme from [16], where a four-stage procedure is proposed. The procedure is hierarchical and one stage must be integral before branching on the next.

First, we branch on the number of vessels used of each vessel class. We define $\hat{a}_v = \sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}} A_{vs}y_{vsw}$ and $v' = \arg\min_{v\in\mathcal{V}} |\hat{a}_v - \lfloor\hat{a}_v\rfloor - 0.5|$. If $\hat{a}_{v'}$ is fractional, we create the branching constraints:

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} A_{v's}y_{v'sw} \leq \lfloor\hat{a}_{v'}\rfloor \tag{27}$$

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} A_{v's}y_{v'sw} \geq \lfloor\hat{a}_{v'}\rfloor + 1 \tag{28}$$

and use them to define the new subproblems. The dual values from these branching constraints are multiplied with the variable a in the objective function of the subproblem of vessel class v'.

Second, we branch on the number of visits to each port by vessels from a given class. We define $\hat{a}_{iv} = \sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}} B_{is}y_{vsw}$ where B_{is} is the number of visits to port i by service s. We let $(i', v') = \arg\min_{(i,v)\in\mathcal{P}\times\mathcal{V}} |\hat{a}_{iv} - \lfloor\hat{a}_{iv}\rfloor - 0.5|$. If $\hat{a}_{i'v'}$ is fractional,

we create the branching constraints:

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} B_{i's}y_{v'sw} \leq \lfloor\hat{a}_{i'v'}\rfloor \tag{29}$$

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} B_{i's}y_{v'sw} \geq \lfloor\hat{a}_{i'v'}\rfloor + 1 \tag{30}$$

and use them to define the new subproblems. The dual value from the branching constraints (29) and (30) is added to all arc variables $x_{i'j}$ and $x_{|\mathcal{P}|+i',j}$ except $x_{i',|\mathcal{P}|+i'}$ and $x_{|\mathcal{P}|+i',i'}$ in the objective function of the subproblem of vessel class v' to encourage/discourage more visits to port i'.

Third, we branch on the total number of times vessels of class v sails from port i to port j. We define $\hat{a}_{ijv} = \sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}} B_{ijs}y_{vsw}$, where B_{ijs} is the number of times service s visits ports i, and j in sequence and $(i', j', v') = \arg\min_{(i,j,v)\in\mathcal{P}\times\mathcal{P}\times\mathcal{V}} |\hat{a}_{ijv} - \lfloor\hat{a}_{ijv}\rfloor - 0.5|$. If $\hat{a}_{i'j'v'}$ is fractional, we create the branching constraints:

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} B_{i'j's}y_{v'sw} \leq \lfloor\hat{a}_{i'j'v'}\rfloor \tag{31}$$

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} B_{i'j's}y_{v'sw} \geq \lfloor\hat{a}_{i'j'v'}\rfloor + 1 \tag{32}$$

and use them to define the new subproblems. The dual value from these branching constraints is added to arc variables $x_{i'j'}$ and $x_{|\mathcal{P}|+i',|\mathcal{P}|+j'}$ in the objective function of the subproblem of vessel class v'.

Branching using the first three strategies is often enough to ensure an integer solution, but there are cases when the solution is fractional even though none of the above strategies indicates a candidate for branching. Figure 4 shows a solution with four services used by vessels from class v. To ease the presentation, we introduce $y_s = \sum_{w\in\mathcal{W}_{vs}} y_{vsw}$. With $y_1 = y_2 = y_3 = y_4 = 0.5$ and assuming that this gives an integer number of vessels used, we see that none of the strategies above gives a candidate for branching.

For the solution not to be integral, there must be fractional y-variables, and among these there must be at least one pair of services with at least one arc in common and at least one arc only present in one service. To have a branching strategy targeting pairs of arcs will therefore detect a candidate for branching and is thus enough to ensure an integer solution. Because of the cyclic nature of the services, the pair can always be constructed such that the tail of one arc is the head of the other arc. In Fig. 4, we see that the arcs (l, i) and (i, k) are not a candidate for branching since both s_1 and s_2 include these arcs. If we instead choose (i, k) and (k, j), only s_1 includes both arcs.

The fourth strategy is therefore to branch on pairs of arcs having one port in com-

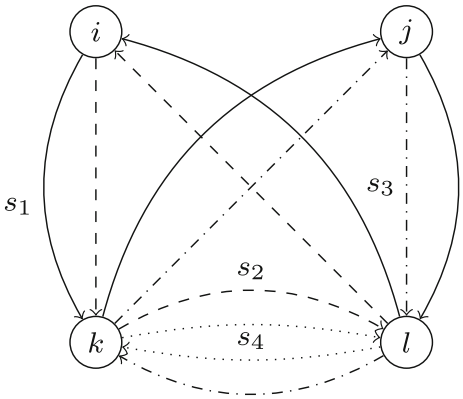

Fig. 4 A solution showing that branching on individual arcs is not sufficient

mon. We define $\hat{a}_{ikjv} = \sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{vs}} B_{ikjs}y_{vsw}$ where B_{ikjs} is the number of times service s visits ports i, k, and j in sequence. Also, we define $(i', k', j', v') = \arg\min_{(i,k,j,v)\in\mathcal{P}\times\mathcal{P}\times\mathcal{P}\times\mathcal{V}} |\hat{a}_{ikjv} - \lfloor\hat{a}_{ikjv}\rfloor - 0.5|$. If $\hat{a}_{i'k'j'v'}$ is fractional, we create the branching constraints:

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} B_{i'k'j's}y_{v'sw} \leq \lfloor\hat{a}_{i'k'j'v'}\rfloor \tag{33}$$

$$\sum_{s\in\mathcal{S}}\sum_{w\in\mathcal{W}_{v's}} B_{i'k'j's}y_{v'sw} \geq \lfloor\hat{a}_{i'k'j'v'}\rfloor + 1 \tag{34}$$

and use them to define the new subproblems for vessel class v'. The dual value from these branching constraints cannot be directly added to the variables in the master problem. Instead, we introduce an integer variable $v_{i'k'j'}$ denoting the number of times the ports i', k', and j' are visited in sequence and add it to the objective function with the dual values from constraint (33) or (34). The following constraints can be added to the subproblem to link the v variables with the x variables:

$$v_{i'k'j'} \leq x_{i'k'} + x_{|\mathcal{P}|+i',|\mathcal{P}|+k'} \tag{35}$$

$$v_{i'k'j'} \leq x_{k'j'} + x_{|\mathcal{P}|+k',|\mathcal{P}|+j'} \tag{36}$$

$$v_{i'k'j'} \leq 2 - x_{k',|\mathcal{P}|+k'} - x_{|\mathcal{P}|+k',k'} \tag{37}$$

$$v_{i'k'j'} \geq x_{i'k'} + x_{k',|\mathcal{P}|+k'} + x_{|\mathcal{P}|+k',|\mathcal{P}|+j'} + x_{|\mathcal{P}|+i',|\mathcal{P}|+k'} + x_{|\mathcal{P}|+k',k'} + x_{k'j'} - 2 \tag{38}$$

$$v_{i'k'j'} \geq x_{i'k'} + x_{k'j'} - 1 \tag{39}$$

$$v_{i'k'j'} \geq x_{|\mathcal{P}|+i',|\mathcal{P}|+k'} + x_{|\mathcal{P}|+k',|\mathcal{P}|+j'} - 1 \tag{40}$$

Since the dual variables corresponding to constraints (33) and (34) are non-positive and non-negative, respectively, only constraints (35), (36), and (37) are needed for for the subproblem created by constraints (33), while constraints (38), (39), and (40) are needed for the subproblem created by constraints (34).

4 Computational Study

The proposed branch-and-price algorithm was implemented in Java SE 8, and all mathematical models are solved using Gurobi Optimizer 9.0.1. The computational study was performed on a computer with 2 x 3.5GHz Intel Xeon Gold 6144 CPU 8 core CPUs and 384 GB RAM. The instances on which the algorithm is tested are presented in Section 4.1. Section 4.2 presents results from preliminary testing where different algorithmic choices are tested and evaluated. The results using this algorithm are compared with best known results and this comparison is made in Section 4.3.

4.1 Test Instances

The instances used in this computational study are taken from [2]. These instances are generated to resemble two common network structures: hub-and-spoke networks and feeder networks (see Fig. 5). The hub-and-spoke networks are usually intercontinental networks where most demand is between different continents. In the instances in [2], the hub-and-spoke networks have two regions with ports. Most of the demands are interregional but there are also demands between ports in the same region. The number of ports in each region is the same. General data about the instances are given in Table 2; for more specific information, see [2].

Each instance is characterized by the structure of the network, the number of ports, and the number of demands. The instances are named S_P_D, where $S = F$ means a feeder network and $S = H$ means a hub-and-spoke network. P is the number of ports and D is the number of demands. Note that in a feeder network, $D = 2(P - 1)$ since all demands either originate or are destined for the hub. This means that the instance H_6_10 is a hub-and-spoke network with six ports and ten demands.

4.2 Preliminary Testing

A subset of six representative instances are used for the preliminary testing. We first test the proposed strengthening of the master problem, constraints (8)–(10). The

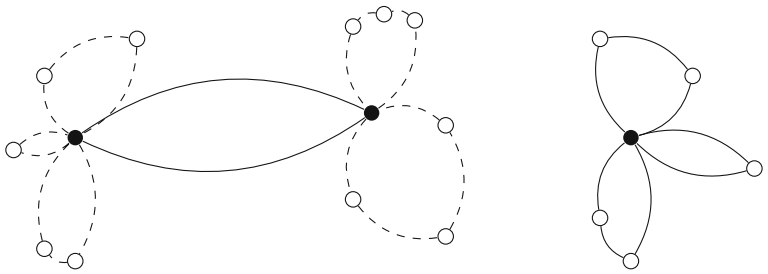

Fig. 5 Left: Hub-and-spoke network with two regions, one hub in each region, one interregional service (solid line), three feeder services in the left region (dashed lines), and two feeder services in the right (dashed lines). Right: Feeder network with three services, the black node is the hub

Table 2 General data about the instances

Characteristics	Feeder	Hub-and-spoke
Avg. demand size	250	250/1000
# vessel classes	2	3
Fixed vessel cost	35'/56'	35'/56'/147'
Capacity	450/800	450/800/2400
Speed (nm/h)	12/14	12/14/16
Consumption (tons/day)	18.8/23.7	18.8/23.7/57.4
Avg. port fee	26'/27'	26'/27'/69'
Avg. transshipment cost	130	130

inner representation of the convex hull of the feasible region defined by capacity constraints over subsets of ports was added to the formulation presented in [2] and the computational study showed that this had a substantial effect on the objective value of the linear relaxation as well as the solution time. All subsets of cardinality up to four, i.e.$|\mathcal{P}_u^S| \leq 4, u \in \mathcal{U}$, are generated and the corresponding constraints are added a priori to the master problem. Table 3 summarizes the results from this first test. We have compared the linear relaxation of four different models; the basic model of [2], Basic, the same model with the inner representation, Best, which was found to give the strongest relaxation in [2], the master problem proposed here, (1)–(6), and the same model strengthened with the inner representation, (1)–(6), (8)–(10). The table shows the relative gap between the linear relaxation and the best solution and is defined as $G_{LP} = 100 \cdot (\overline{z}_{IP} - z_{LP})/(\overline{z}_{IP} - z_{LP}^{BASIC})$, where z_{LP} is the objective value of the linear relaxation, z_{LP}^{BASIC} is the objective value of linear relaxation of the basic model in [2], and $\overline{z}_{IP}$ is the best objective value found in [2].

First, we note that the linear relaxation of the master problem is stronger than the linear relaxation of the formulation with the inner representation as presented in [2] on the three largest instances. We also see a clear improvement in the objective value of the master problem with the inner representation, the mean remaining gap decreases from 39.1 to 18.2%. Based on this preliminary testing, we decide

Table 3 A comparison of the relative gaps for the different linear relaxations

	Ameln et al. [2]		This work	
Instance	Basic	Best	(1)–(6)	(1)–(6), (8)–(10)
*F*_4_6	100.0	27.7	34.1	17.2
*F*_8_14	100.0	29.7	27.7	17.6
*F*_12_22	100.0	37.7	25.9	19.0
*H*_4_6	100.0	44.0	53.6	22.4
*H*_6_8	100.0	34.6	61.0	18.7
*H*_8_10	100.0	52.0	39.4	17.2

to continue the computational study with the formulation including the proposed strengthening of the master problem. This means that we use the models (1)–(6), and (8)–(10) unless otherwise stated.

Finding the column with the most negative reduced cost is not necessary in each iteration, instead we only need one column with negative reduced cost for the RMP to produce a new dual solution. Hopefully, the full subproblem only needs to be solved once for every vessel class in every node, to verify that there are no more columns with negative reduced cost and thus that the node is solved to optimality. Table 4 summarizes the results from including no acceleration strategies, No acc., including the improvement of the delivery patterns, Improve, and including both strategies, Both acc., with respect to the number of columns generated, while Table 5 focuses on the time spent.

Table 4 shows the number of columns generated by the different strategies in the three different settings tested. C^{IP}, C^{LP}, and C^{LS} are the numbers of columns found by the full subproblem, by the improve delivery pattern strategy, and by the local search strategy respectively. I^{IP} is the number of times the full subproblem is called. When no acceleration strategies are used, the full subproblem generates on average 1.8 columns per call, while this number decreases to 0.4 when both strategies are used. This is a clear indication that the full subproblem is mainly verifying optimality and that the acceleration strategies find many columns. Ideally, $C^{IP} = 0$ when acceleration strategies are introduced. We do not reach this, the only exception is F_4_6, but there is a substantial decrease in the number of columns generated by the full subproblem and the number of times the full subproblem is solved.

The effect of this decrease on the solution time is shown in Table 5. Here, T^{IP}, T^{LP}, and T^{LS} are time spent in the full subproblem, in the improve delivery pattern strategy, and in the local search strategy, respectively. T is the total time spent in the algorithm. We see a dramatic decrease in solution time by including the acceleration strategies for the instances that are solved.

Finally, Table 6 summarizes the preliminary testing and shows the gap, G, solution time, T, and number of searched nodes, N, from the testing of the acceleration strategies. We define the relative gap between the lower bound after 1 h and the best solution found in [2] as $G = 100 \cdot (\overline{z}_{IP} - z_{LB})/(\overline{z}_{IP} - z_{LP}^{BASIC})$, where z_{LB} is the lower bound after 1 h. The lower bound after 1 h and the solution time of the model

Table 4 Results of the proposed acceleration strategies with respect to the number of columns generated

	No acc.		Improve			Both acc.			
Instance	C^{IP}	I^{IP}	C^{IP}	C^{LP}	I^{IP}	C^{IP}	C^{LP}	C^{LS}	I^{IP}
F_4_6	411	257	36	225	50	0	220	57	15
F_8_14	4527	995	1658	29816	935	747	45,891	3410	506
F_12_22	2233	192	343	17,327	56	116	6700	1624	28
H_4_6	686	1822	60	453	1141	6	414	61	1030
H_6_8	10,706	8198	629	5395	2556	166	4871	469	2080
H_8_10	4211	1131	2109	16070	2146	1094	22930	3997	1781

Table 5 Results of the proposed acceleration strategies with respect to the time spent

	No acc.		Improve			Both acc.			
Instance	T^{IP}	T	T^{IP}	T^{LP}	T	T^{IP}	T^{LP}	T^{LS}	T
F_4_6	28	29	4	0	4	1	0	0	1
F_8_14	3557	3600	3182	38	3600	2539	68	305	3600
F_12_22	3486	3600	2661	28	3600	2780	11	615	3600
H_4_6	152	161	32	1	35	29	1	0	31
H_6_8	2736	3019	360	20	456	315	23	22	429
H_8_10	3498	3600	3221	81	3600	2648	163	360	3600

in [2] with the inner representation is added as a comparison. Note that for instance F_12_22, the root node is not solved within 1 h. No method is clearly better than the other, [2] solves instance F_8_14 while the relative gap on instance H_8_10 is much higher. We see a clear positive effect when comparing the acceleration strategies; using both strategies gives much shorter solution times on the instances that are solved and a much higher number of nodes, and thus a better relative gap, on the instances that are not solved.

4.3 Comparison and Results

One of the main purposes with the Dantzig-Wolfe reformulation is to improve the dual bound. In Section 4.2, we saw that the lower bounds at the root node produced with the proposed formulation are clearly better than the corresponding bounds presented in [2]. Here, we present a more comprehensive study where all instances are included, and the computational time is set to 1 h. In Table 7, we present a comparison between results from the algorithm proposed here and results by the method proposed in [2]. The table shows the name of the instance, the root node gap calculated as $G_{LP} = 100 \cdot (\overline{z}_{IP} - z_{LP})/\overline{z}_{IP}$, the final gap calculated as $G = 100 \cdot (\overline{z}_{IP} - z_{LB})/\overline{z}_{IP}$,

Table 6 Summarized results of the proposed acceleration strategies

	Best		No acc.			Improve			Both acc.		
Instance	G	T	G	T	N	G	T	N	G	T	N
F_4_6	0.0	2	0.0	29	5	0.0	4	5	0.0	1	5
F_8_14	0.0	1809	17.6	3600	1	5.1	3600	21	5.0	3600	20
F_12_22	29.3	3600	–	3600	0	–	3600	0	–	3600	1
H_4_6	0.0	5	0.0	161	375	0.0	37	363	0.0	32	353
H_6_8	0.0	126	0.0	3018	594	0.0	456	637	0.0	429	641
H_8_10	27.6	3600	15.7	3600	8	9.1	3600	180	8.1	3600	256

Table 7 Comparison of the gaps between the method presented in [2] and the proposed algorithm

	Ameln et al. [2]			Branch-and-price		
Instance	G_{LP}	G	Time	G_{LP}	G	Time
F_4_6	12.8	0	1	8.0	0	1
F_6_10	5.2	0	4	0	0	40
F_8_14	14.4	0	288	8.5	2.1	3600
F_10_18	14.4	9.0	3600	7.5	7.4	3600
F_12_22	26.2	22.4	3600	17.0	17.0	10750*
H_4_4	11.2	0	2	1.5	0	3
H_4_6	13.6	0	4	6.9	0	34
H_4_8	13.5	0	5	4.8	0	16
H_6_6	12.5	0	50	7.2	0	108
H_6_8	11.4	0	30	6.1	0	404
H_6_10	12.0	0	35	4.2	0	23
H_6_12	16.0	0	1924	9.8	4.6	3600
H_6_14	20.8	0	783	15.2	7.7	3600
H_6_16	14.9	0	1775	8.0	2.6	3600
H_8_8	21.3	2.2	3600	10.3	5.1	3600
H_8_10	18.5	4.5	3600	5.9	2.5	3600
H_8_12	21.9	14.8	3600	9.8	6.9	3600
H_8_14	17.4	6.3	3600	8.7	6.0	3600
H_8_16	22.5	17.8	3600	16.7	13.3	3600
H_10_10	18.8	10.6	3600	9.4	5.7	3600
H_10_12	26.9	20.4	3600	11.8	10.2	3600
H_10_14	25.7	20.3	3600	17.2	15.8	3600
H_10_16	25.2	19.1	3600	15.3	14.0	3600
Average	17.3	6.4	1935	9.1	5.3	2375

and the computational time for both algorithms. Note that the same upper bound is used for both algorithms. Instances that are solved to proven optimality are marked with a 0 in the G column.

We see that the method proposed in [2] solves 12 instances while our method solves eight. On the other hand, the average relative gap at the root node is almost halved, 17.3 compared with 9.1, in favor of our method and the average relative gap after 1 h is also smaller, 6.4 compared with 5.3. This improvement is especially clear on the larger instances, $|\mathcal{P}| \geq 10$, where the relative gaps are 22.9 and 17.0 compared with 13.0 and 11.7. Note that instance F_12_22 is run until the root node is solved, the reported result is the root node solution and the time to solve the root node. This has been marked with an asterisk in the table. In [2], the authors report the gap of F_12_22 after 10 h to be 21.6, clearly worse than the root node gap reported here.

5 Conclusions

It is clear that the liner shipping network design problem is a notoriously hard problem. The transshipment possibilities decouple the loading and unloading decisions, and clearly weaken the formulation. Many other routing problems use dynamic programming for solving the pricing problem, but this is not a suitable method here. A reason for this is that you do not have a node with a well-defined state from where to start the algorithm. Often a depot is defined and the fact that the vehicle is either empty when leaving the depot or returning to it is used to start the algorithm. Another reason is transshipment, which creates a large number of loading and unloading possibilities in each node.

We find that capacity cuts defined as an inner representation of the convex hull of the feasible region from [2] significantly strengthen the master problem. The resulting LP relaxation of our model is considerably stronger than that of [2], giving an improved initial dual bound at the cost of computational time needed to solve the root node of the branch-and-bound tree. We see that closing the gap is challenging, and we are not able to reach optimality within 1 h for instances larger than six ports and ten demands.

There is still room for improvements within the approach taken here, and results could likely be improved by examining more strategies in terms of branching, branch-and-bound tree traversal, improved handling of columns in the master problem, and better heuristic methods for finding columns. However, there are limits to what can be done with algorithm engineering, and significant time is spent in the relatively complex full subproblem. This needs to be solved at least once in each fully explored branch-and-bound node to prove optimality, and makes the method impractical for finding optimal solutions to instances of any significant size. It is clear that heuristic methods are needed to find good primal solutions, but in terms of dual bounds we have found an improvement over previously published results on the same instances, especially for larger instances.

Acknowledgments We are grateful to the reviewers, whose comments helped us improve the paper.

Funding Open Access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital). This work was partly supported by the Research Council of Norway through the AXIOM project. This support is gratefully acknowledged.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Agarwal R, Ergun Ø (2008) Ship scheduling and network design for cargo routing in liner shipping. Transp Sci 42:175–196
2. Ameln M, Fuglum JS, Thun K, Andersson H, Stålhane M A new formulation for the liner shipping network design problem. Int Trans Oper Res 2019. https://doi.org/10.1111/itor.12659
3. Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MPW, Vance PH (1998) Branch-and-price: column generation for solving huge integer programs. Oper Res 46(3):316–29
4. Brouer BD, Alvarez JF, Plum CEM, Pisinger D, Sigurd MM (2014) A base integer programming model and benchmark suite for liner-shipping network design. Transp Sci 48(2):281–312
5. Christiansen M, Fagerholt K, Nygreen B, Ronen D (2012) Ship routing and scheduling in the new millennium. Eur J Oper Res 228(3):467–483
6. Christiansen M, Hellsten E, Pisinger D, Sacramento D, Vilhelmsen C (2020) Liner shipping network design. European J Oper Res 286(1):1–20. https://doi.org/10.1016/j.ejor.2019.09.057
7. Desaulniers G (2010) Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Oper Res 58(1):179–192
8. Desrosiers J, Lübbecke ME (2005) A primer in column generation. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Column generation. Springer, New York, pp 1–32
9. Lawrence SA (1972) International sea transport: the years ahead. Lexington Books, Lexington
10. Lee C-Y, Song D-P (2017) Ocean container transport in global supply chains: overview and research opportunities. Transport Res Part B Meth 95:442–474
11. Meng Q, Wang S, Andersson H, Thun K (2014) Containership routing and scheduling in liner shipping: overview and future research directions. Transp Sci 48(2):265–280
12. Plum CEM, Pisinger D, Sigurd MM (2014) A service flow model for the liner shipping network design problem. European J Oper Res 235(2):378–386
13. Reinhardt LB, Pisinger D (2011) A branch and cut algorithm for the container shipping network design problem. Flex Serv Manuf J 24(3):349–374
14. Santini A, Plum CEM, Stefan Ropke A (2018) Branch-and-price approach to the feeder network design problem. European J Oper Res 264(2):607–622
15. Stålhane M, Andersson H, Christiansen M, Cordeau JF, Desaulniers G (2012) A branch-price-and-cut method for a ship routing and scheduling problem with split loads. Comput Oper Res 39:3361–3375
16. Thun K, Andersson H, Christiansen M (2017) Analyzing complex service structures in liner shipping network design. Flex Serv Manuf J 29(3):535–552
17. Tran NK, Haasis HD (2015) Literature survey of network optimization in container liner shipping. Flex Serv Manuf J 27(2-3):139–179
18. UNCTAD Review of maritime transport 2019. Technical report, United Nations