Integrated Liner Shipping Network Design and Scheduling

Koza, David Franz; Desaulniers, Guy; Røpke, Stefan

Published in:
Transportation Science

Link to article, DOI:
[10.1287/trsc.2018.0888](https://doi.org/10.1287/trsc.2018.0888)

Publication date:
2020

Document Version
Peer reviewed version

[Link back to DTU Orbit](#)

Citation (APA):
Koza, D. F., Desaulniers, G., & Røpke, S. (2020). Integrated Liner Shipping Network Design and Scheduling. *Transportation Science*, 54(2), 512-533. <https://doi.org/10.1287/trsc.2018.0888>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Integrated Liner Shipping Network Design and Scheduling

D.F. Koza, G. Desaulniers, S. Ropke

May 15, 2019

Abstract In global liner shipping networks a large share of transported cargo is transshipped at least once between container vessels, and the total transportation time of these containers depends on how well the corresponding services are synchronized. We propose a problem formulation that integrates service scheduling into the liner shipping network design problem. Furthermore, the model incorporates many industry-relevant modeling aspects; it allows for leg-based sailing speed optimization, it is not limited to simple or butterfly-type services and it accounts for service level requirements like cargo transit time limits. The classic liner shipping network design problem is already a hard problem and in order to solve the extended version we propose a column generation matheuristic that uses advanced linear programming techniques. The proposed method solves LINER-LIB instances of up to 114 ports and, if applied to the classic liner shipping network design problem, finds new best solutions to all instances, outperforming existing methods reported in the literature. Additionally, we analyze the relevance of scheduling for liner shipping network design. The results indicate that neglecting scheduling and approximating transshipments instead may result in the design of liner shipping networks that underestimate cargo transit times and its implications.

Introduction

Containerization allows to seamlessly move cargo between different modes of transportation and is considered as one of the strongest drivers of international trade (Bernhofen et al., 2016). The backbone of the global trade network is formed by overseas cargo carriers that transport around 70% of the world seaborne trade in value (WTO, 2008). By offering comparatively very low transportation cost, container shipping networks have actually made today’s global sourcing and production networks possible (Notteboom, 2012).

The largest carriers in the highly concentrated liner shipping market run several hundred vessels on more than a hundred services. Each service represents a sailing route that is regularly operated following a published schedule. Liner services are connected through common port calls that allow liner network operators to move cargo from one service to another. The movement of containers between services is called transshipment and enables large cargo carriers to transport containers between almost any possible pair of ports around the globe.

The question of how to redesign a liner shipping network is at the core of optimization problems faced by cargo carriers and involves various trade-offs. While shippers generally request eco-

nomic, fast, and ideally direct port-to-port connections, cargo carriers try to achieve high vessel capacity utilization and low operational costs. In order to achieve economies of scale, the average container vessel capacity has almost doubled from 1,732 twenty-foot equivalent units (TEU) to 3,394 TEU between 2000 and 2014. The capacity of the largest vessels has quadrupled since the Panamax era in the mid-1990s (Tran and Haasis, 2015a), with the largest vessels being able to carry more than 20,000 TEU in 2018. On the downside, ultra-large container vessels (ULCV) can only call selected ports and therefore offer less direct port-to-port connections. Regional services operated by smaller vessels are commonly used to collect regional cargo and feed the ULCV at large hub ports. Indeed, the share of transshipped containers among global container port throughput has increased from 11% to 28% between 2008 and 2012 (Drewry, 2013). Today, in the network of Maersk Line, the world’s largest overseas cargo carrier, more than half of all transported containers are transshipped at least once before arriving at their final destination.

The integration of network design and scheduling decisions comes along with important practical benefits. A large share of transshipped containers requires liner services to be well synchronized. Whether services are well synchronized is determined by their individual schedules, because these determine the transshipment times of cargo between services and, ultimately, the transit times of cargo. Moreover, the schedules determine the buffer times between port calls. With vessel delays being quite common, buffer times have a crucial impact on a carrier’s reliability for both its schedule and cargo transit times. High transit time reliability and schedule reliability constitute a competitive advantage and justify higher transport rates (Notteboom, 2006; Vernimmen et al., 2007). Finally, the schedule of a liner shipping service implicitly defines the vessels’ sailing speeds and thus the bunker consumption per sailing leg, which represents the largest operational cost for cargo carriers.

In practice, large cargo carriers do not redesign their whole network from scratch. Gradual modifications, however, do occur frequently, and the networks change constantly; reschedulings of single port calls or adjustments of single services within a global network occur on a weekly to monthly basis. Larger modifications of the network, such as the redesign of multiple services within a region, are often made on an annual basis and involve large numbers of vessels and ports. The 2M alliance between Maersk Line and MSC, for example, operates more than 150 vessels on more than 20 services and revises its shared network at least once per year. Due to the strong connectivity of liner shipping networks, the synchronization of both the redesigned and the remaining part of the network are critical for a cargo carrier’s competitiveness.

We present a mathematical model and solution method for the integrated Liner Shipping Network Design and Scheduling Problem (LSNDSP). Our model and solution method cover many practically relevant aspects which, to the best of our knowledge, have never been combined into

a single model. The proposed method finds solutions of high quality for LINER-LIB instances of up to 114 ports. To summarize, the contributions are:

- We introduce the first mixed integer programming model for the integrated LNSDSP.
- Transshipment times are modeled in an exact way and not approximated by a constant. Therefore, transit times are also exact with respect to the discretization used.
- The proposed model covers many industry-relevant aspects: leg-based sailing speed optimization, consideration of service level requirements, and the modeling of any type of service rotation.
- We develop a novel column generation matheuristic that combines linear programming techniques with heuristics. The method can be used to construct new liner networks or to extend or improve existing liner networks.
- We report results for instances of the LINER-LIB benchmark suite involving up to 114 ports. When applied to the classic liner shipping network design problem, the method finds new best solutions for all considered LINER-LIB instances.

The paper is organized as follows. Section 1 reviews related literature. In Section 2 we state and describe the LNSDSP. In Section 3 we first define a rich graph model (Section 3.1). Based on this model, we then state a mathematical formulation of the problem (Section 3.2) and shortly outline similarities and differences with other service network design models. In Section 4 we describe a column generation based solution algorithm and discuss relevant implementation details. These include a simple but highly efficient dual variable penalization technique and an improved solution algorithm for the time constrained multi-commodity flow problem. In Section 5 we present computational results based on the public LINER-LIB data instances. A comparison with results from the literature shows the effectiveness of the proposed solution method. Finally, concluding remarks and directions for future work are given in Section 6.

1 Literature review

The LNSDSP contains many other studied maritime optimization problems as subproblems, among these the ship routing and scheduling problem, the fleet deployment problem, the speed optimization problem and the cargo flow problem. Researchers have addressed variations and combinations of these subproblems in different contexts. Guericke and Tierney (2015), for example, assume a given network design and vessel deployment, but allow for simultaneous vessel speed optimization and cargo allocation. A similar problem is addressed by Akyüz and Lee (2016); the service network design and the available fleet of vessels are given, but the type, number and average speed of vessels on each service is part of the decision problem. The work

by Plum et al. (2014a) combines service design and cargo routing decisions for a single liner shipping service that has to connect a predefined set of ports.

The body of literature that addresses one or multiple of the above mentioned maritime optimization problems is vast and we refer the reader to the reviews by Christiansen et al. (2013), Meng et al. (2014) and Tran and Haasis (2015b) for an overview of optimization problems within liner shipping. In the following we focus on liner shipping *network* design studies that explicitly design the shipping services and consider transshipments of containers between those services.

The work by Agarwal and Ergun (2008) is regarded the first on liner shipping network design as it is defined and distinguished from other maritime routing problems. In particular, they were the first to consider a weekly service frequency and the possibility of cargo transshipments between services. Their model, however, does not account for transshipment costs. It is noteworthy that the model by Agarwal and Ergun (2008) also determines schedules for each service. It is a feature that most of the subsequent publications on liner shipping network design neglect. For better distinction from the LSNDSP we will use the acronym ULSNDP for unscheduled liner shipping network design problems in the remainder of the paper.

Álvarez (2009) presents a two-tier solution approach for the ULSNDP based on tabu search and column generation that is tested on a case study consisting of 120 ports. Gelareh and Pisinger (2011) propose a mixed integer programming formulation for regional liner shipping hub-and-spoke network design, together with a solution method based on Benders' decomposition. In order to promote research on liner shipping network design problems and to facilitate the comparison of solution approaches, Brouer et al. (2014a) develop a benchmark suite (LINER-LIB) and give an extensive introduction into the domain of liner shipping network design. They present a solution algorithm that is an extension of the heuristic by Álvarez (2009) and impose the requirement of (bi)weekly service frequencies. Another iterative heuristic for the ULSNDP was developed by Mulder and Dekker (2014) to solve a network design problem instance on the Asia-Europe trade lane. Brouer et al. (2014b) present a matheuristic to solve the ULSNDP. The proposed method is an improvement heuristic with an integer program at its core that is used to iteratively modify existing services and re-evaluate the network by solving cargo flow problems.

Brouer et al. (2015) and Karsten et al. (2017a) extend the work of Brouer et al. (2014b) by additionally considering restrictions on the total transit time and on the number of transshipments for each origin-destination container path. Karsten et al. (2017b) further extend the matheuristic by allowing varying speeds per leg on each service. The main contribution of this series of papers is the integration of service levels and thus of shippers' requirements and preferences into the liner shipping network design problem. Besides the pure transportation cost, the expected transit time and the number of transshipments of cargo are important decision criteria

Figure 1: Illustration of different service types

for shippers. Results are presented for all but the largest LINER-LIB instances, including large global liner networks. The service schedules are, however, considered to be independent and transshipment times are approximated by a constant of 48 hours.

Meng and Wang (2011) propose a mixed integer programming (MIP) formulation for a network design problem that additionally considers empty container repositioning to maintain a balance between container demand and supply at each port. Their solution method, however, does not design services, but selects services from a given set. Results are presented for randomly generated instances of 46 ports and a maximum of 60 shipping lines to select from. Balakrishnan and Karsten (2017) address a similar problem. They consider multiple liner shipping services and resulting network effects, but do not explicitly design the services; their solution approach selects the best combination among a predefined pool of services and simultaneously determines the routing of cargo.

Although heuristic solution approaches are dominant, a few exact algorithms for the ULSNDP have been proposed as well. Reinhardt and Pisinger (2012) present a compact formulation of the ULSNDP with transshipments, but services are not required to satisfy a weekly frequency. The model contains a large number of constraints, including *big-M* constraints, and the proposed branch-and-cut algorithm can solve to optimality only small instances with up to 10 ports and 12 origin-destination demands.

An alternative compact formulation of the ULSNDP is proposed by Plum et al. (2014b) and applied to the two smallest instances of the LINER-LIB benchmark suite. An optimality gap remains for both instances. The main contribution is the model’s capability to represent services of any type, i.e. services can visit any port multiple times. These types of services, which we denote *complex* in distinction to *simple* and *butterfly* type services (see Figure 1), are very common in practice.

Thun et al. (2017) analyze the potential cost savings that can be obtained by allowing complex services. Their study is motivated by the fact that the vast majority of models and solution methods is limited to simple and butterfly type services. Even though the considered instances

are small (up to 7 ports and 14 demands), the results of their branch-and-price algorithm indicate that the use of complex services may allow for savings compared to networks solely consisting of simple and butterfly type services.

A stream of research related to the LSNDSP addresses models and solution methods for *service network design* in freight transportation. Note that in the service network design literature, “service” usually denotes a transportation service between e.g. terminals, whereas a service in liner shipping represents a whole sequence of port-to-port legs. Service network design problems are a generalization of network design problems and address tactical decision problems faced by freight carriers. Service network design problems are primarily characterized by their functionality rather than by the mode of transportation. The decisions that characterize service network design problems are the selection and scheduling of transportation services and the routing of cargo. Commonly, individual origin-destination demands are too low in service network design problems to be transported efficiently without consolidation. Crainic (2000) provides a classification of service network design problems and reviews modeling and solution approaches. The model we propose for the LSNDSP is a variation of a service network design model and shares similarities with scheduled cyclic service network design models as presented by Andersen et al. (2011) and Crainic et al. (2016). In Section 3 we will discuss differences between the LSNDSP and existing service network design problems and models in more detail.

2 Problem description

The LSNDSP consists of two distinct but interdependent problems that are addressed simultaneously: On one hand, given an available fleet of vessels of different types, the decision maker has to define a set of services to be operated. On the other hand, the decision maker has to decide which origin-destination demand to transport and how to route it through the network. While the operation of services implies costs, the transportation of cargo generates revenues. Evidently, both problems are interdependent: the design and schedule of the network is driven by the potential revenues to gain, whereas the cargo routing options and transit times are defined by the network and schedule design.

A service represents a cyclic sequence of scheduled port calls and is operated by vessels of the same vessel class. The total round trip time of a service is equal to the sum of speed-dependent sailing times between served ports and a port stay time for each port call, covering loading and unloading operations. In practice, the port stay time depends on the number of container loading and unloading moves, the port’s productivity, and the number of cranes assigned to the vessel, among others; terminal operators, however, typically assign more resources to larger vessels with high move counts, as they take up more space. We therefore adapt the assumption of constant

(a) Maersk Line Meso American Express round trip (b) Maersk Line Amazonia Caribbean Express round trip

Figure 2: Two examples of Maersk Line services (Maersk Line, 2017). The ports of Cartagena, Colombia, and Manzanillo, Panama, are called by both services and allow cargo to be transshipped between the two services. Note that both services are complex services.

port stay times for every port call, as used in previous works on liner shipping network design.

Figures 2a and 2b show two different Maersk Line services of total durations of 5 and 9 weeks, respectively. In the following we assume a weekly frequency for all services as predominant in the liner shipping industry. In order to ensure a weekly service frequency, the number of vessels deployed on a service has to be equal to the duration of the service in weeks. The maximum duration of a service is limited by the number of available vessels per vessel class.

A vessel class is defined by its capacity in forty-foot equivalent units (FEU), a minimum and maximum speed, and by its ship measurements. A vessel's dimension or draft may disqualify the vessel class for some ports or canals and a vessel class' speed limitations may not allow it to operate a particular schedule. The cost of operating a vessel of a particular class consists of a (weekly) charter rate and sailing speed dependent fuel cost.

A demand represents a weekly quantity of FEU for a particular origin-destination pair of ports. Each demand is associated with a unit revenue. Additionally, transit time limits may apply, reflecting a demand's time-sensitivity. The transit time limits may arise from the fact that transported cargo is perishable, but typically they reflect market requirements towards competitive transit times. Demands of different revenues and time-sensitivities may exist for the same origin-destination pairs. A limit on the number of transshipments may be defined for a demand, reflecting the shippers preference towards direct shipping routes. A demand can, but does not have to be served by the cargo carrier. Moreover, it can be fulfilled partially.

Cargo can generally be transshipped between services, but every transshipment implies addi-

tional cargo handling costs which may vary between ports. The transshipment time between two services depends on the schedules of the unloading and the loading service. If a minimum transshipment time is not met, cargo may have to wait one week for the next vessel to arrive. A minimum transshipment time of containers is typically required for unloading, storage, transport between the quays, and loading. We chose to not consider the possibility of 'hot transshipments', i.e. the simultaneous transshipment of containers in both directions between two vessels that berth at the same terminal, even though the model we propose could handle them. The reason is that the feasibility of hot transshipments is subject to specific conditions of which some have to be negotiated with the terminals, as e.g. the availability of two berthing slots in close distance at the same terminal and time.

The objective of the LSNDSP is to design and schedule a network of services such that the (weekly) cost of operating the services minus the earned revenue for serving demands is minimized. The cargo carrier cannot use more than the available number of vessels for each vessel class and has to satisfy transit time or transshipment restrictions for all served demands.

The liner shipping network design problem and the service scheduling problem have traditionally been addressed independently. Classic liner shipping network design solution approaches aim at determining a set of cyclic routes, with each route being defined as a sequence of ports to visit. Recent liner shipping network design approaches additionally optimize the sailing speed on individual legs; resulting solutions determine the sailing time between pairs of port calls, but they do not determine arrival and departure times of vessels at ports. As a consequence, transshipment times cannot be determined and are usually approximated by a constant. The integration of scheduling into classic liner shipping network design determines arrival and departure times for all services and ports. This additional information allows to determine transshipment times between services and thus the total transit time of any cargo origin-destination path. The integration of scheduling, however, comes at the cost of a drastically increased solution space.

3 A model for the LSNDSP

The model we propose for the LSNDSP is formulated over a directed time-space graph. We first describe the graph modeling idea and define resulting graphs in Section 3.1. The mathematical model of the LSNDSP is presented in Section 3.2.

3.1 Graphs

We model the problem over a directed time-space graph, which allows to not only model varying sailing speeds between subsequent port calls, but also schedule dependent transshipment times between different services that call the same port. The (weekly) time is discretized into uniform

Figure 3: Illustration of a time-space graph G . Solid arcs represent sailings between ports and dashed arcs represent transshipments. The colored arcs denote examples of cargo paths; the blue path from port D to port C is direct, whereas the red path from port A to port D requires a transshipment at port C. Backward arcs represent sailings or transshipments that end in a subsequent week.

steps of length $\bar{h}$ (a divisor of 168 hours), represented by the set $H = \{0, 1 \cdot \bar{h}, 2 \cdot \bar{h}, \dots, 168 - \bar{h}\}$. Let P denote the set of ports and L the set of legs, representing all feasible port-to-port sailings. Set L may be larger than $|P| \times |P|$ because for some pairs of ports different sailing routes may exist (e.g. one using Suez canal and another one sailing around Africa).

All feasible services and all feasible cargo paths can be represented in a basic directed graph, denoted $G = (V, A)$, where V and A are its vertex set and arc set, respectively. Set V denotes ports at a particular departure time and contains $|P| \times |H|$ vertices, namely, one vertex for each port $i \in P$ and each time $h \in H$. A vertex $(i, h) \in V$ represents a departure from port i at time h . Figure 3 illustrates a small time-space graph with two services (each with a one-week duration) and two cargo paths.

The arc set A connects some of the vertices to each other and is the union of two arc sets A^S and A^T . Arc set A^S represents *sailing arcs* and connects two port-time vertices (i_1, h_1) and (i_2, h_2) with $i_1 \neq i_2$ to each other, if a vessel class exists that can sail a leg $l \in L$ from port i_1 to port i_2 departing at time h_1 such that it is ready for departure from port i_2 again at time h_2 . The duration t_a of an arc $a \in A^S$ includes the sailing time between ports (possibly including waiting time outside of the destination port) and the port stay time at the destination port. Note that the one-week time horizon is cyclic, which means that all times should be computed modulo 7 days (168 hours). Let us, for example, consider a sailing leg from port i_1 to port i_2 departing at time $h_1 = 8$: If the minimum and maximum sailing times between ports i_1 and i_2 are 128 and 152 (all times in hours), respectively, the port stay time is 24 and a time discretization of $\bar{h} = 8$ is used, the earliest weekly departure time from port i_2 after this leg is $(8 + 128 + 24) \bmod 168 = 160$ and the latest weekly departure time is $(8 + 152 + 24) \bmod 168 = 16$. Hence,

the corresponding sailing arcs would connect vertex $(i_1, 8)$ to vertices $(i_2, 160)$, $(i_2, 0)$, $(i_2, 8)$ and $(i_2, 16)$. The solid arcs in Figure 3 represent selected sailing arcs, including two arcs representing the same sailing (between ports B and C) but at different speeds. Note that, by definition, all cycles in graph $G = (V, A)$ have a duration equal to a multiple of one week (168 hours).

The second subset of arcs, *transshipment arcs* A^T , represents the transshipment of cargo from one service to another at a port $i \in P$ with a particular layover time. Transshipment arcs exist between all port-time vertices (i_1, h_1) and (i_2, h_2) with $i_1 = i_2$. The dashed arcs in Figure 3 illustrate transshipments. The duration t_a associated with a transshipment arc $a \in A^T$ depends on the minimum layover time necessary for a transshipment at a particular port. Note that the layover time is defined as the time between arrival of the unloading vessel and the departure of the loading vessel. Hence, given a port stay time of 24 and a minimum layover time of 48, the resulting outgoing transshipment arcs of e.g. vertex $(i_1, 8)$ are of duration $48 - 24 = 24$ to $160 + 48 - 24 = 184$ and thus connect to all other vertices of the same port, $(i_1, 32), (i_1, 40), \dots, (i_1, 160), (i_1, 0), \dots, (i_1, 24)$, stated in the order of increasing arc duration t_a .

In theory, there can be $|P| \times |P - 1| \times |H|^2$ arcs between vertices $v, u \in V$. This number is even larger if parallel arcs exist between port-time vertices due to the same sailing at different speeds arriving a multiple of weeks apart (i.e. same sailing (i_1, h_1) to (i_2, h_2) , but different arc durations t_a that differ by a multiple of one week). These cases may occur, as e.g. a sailing between Europe and the U.S. east coast can take a week or two, depending on the sailing speed. In order to not overload notation, we will denote in the remainder of the paper an arc $a = (u, v)$ by its origin and destination vertices u and v , assuming a unique duration t_a for every pair (u, v) . Our implementation and computational tests, however, account for parallel arcs of different durations. Graph G contains all vertices and arcs that are feasible for at least one vessel class and cargo path.

Different port restrictions as well as varying vessel-dependent feasible ranges of speeds make different vertices and arcs of graph G feasible for each vessel class. We denote the set of vessel classes by E . Let $G_e = (V_e, A_e)$ denote the subgraph that contains all vertices and arcs feasible for vessel class $e \in E$. Set $V_e \subseteq V$ contains all vertices corresponding to time-copies of ports that can be visited by a vessel in class $e \in E$. Set $A_e^S \subseteq A^S$ contains all sailing arcs that are feasible for a vessel in class $e \in E$, where feasibility of an arc $a = (v, u)$ depends on whether $v, u \in V_e$ and whether the sailing time associated with arc a is feasible for this vessel class. For each vessel class, sailing leg and departure time we add one arc that represents the slowest feasible speed plus a minimal idle time (≥ 0) in order to guarantee that, despite the time discretization, the slowest feasible sailing speed is modeled in the graph. Note that arc set A_e does not contain any

transshipment arcs. Sets $A_e^+(v)$ and $A_e^-(v)$ are used to denote outgoing and incoming arcs of vertex $v \in V$ that are feasible for vessel class $e \in E$. We use the notation $E_v \subseteq E$ and $E_a \subseteq E$ to denote the vessel classes that are allowed to use vertices $v \in V$ and arcs $a \in A^S$, respectively.

Different vertices and arcs may be feasible for different demands, which we denote by K . Feasible paths for a demand $k \in K$ with origin O_k and destination D_k correspond to paths in G that start at a vertex $v \in V$ whose associated port is equal to O_k and end at a vertex $v \in V$ whose associated port corresponds to D_k . We define subgraphs $G_k = (V_k, A_k)$ for each demand $k \in K$ that contain all these paths. Set $V_k \subseteq V$ contains all vertices $v \in V$ whose associated port can be part of a feasible path from O_k to D_k of demand k while respecting the demand maximum transit time. Set $A_k \subseteq A$ contains all arcs in A linking vertices in V_k , except all arcs that would be infeasible with respect to the maximum transit time, and all arcs associated with a sailing ending in O_k or starting in D_k . Accordingly, we use the notation $K_v \subseteq K$ and $K_a \subseteq K$ to denote the demands that are allowed to use vertices $v \in V$ and arcs $a \in A$, respectively.

3.2 Mathematical model

In this section, we provide a MIP formulation for the LSNDSP. The LSNDSP is formulated on graph $G = (V, A)$. The goal of the model is to design services and to select cargo paths that use the services while minimizing total cost minus total revenue.

The binary decision variables y_a^e indicate whether a sailing leg and speed represented by arc $a \in A^S$ is sailed by vessel class $e \in E$ ($y_a^e = 1$) or not ($y_a^e = 0$). The set Ω contains all feasible cargo paths; it can be split into disjoint sets Ω_k , one for each demand $k \in K$. The continuous decision variables x_j^k denote the number of containers of demand $k \in K$ that are transported along path $j \in \Omega_k$. While the number of transported containers can only be integer, the error resulting from relaxing the integrality requirements on these decision variables is negligible in practice given that vessels can carry several thousand containers (Brouer et al., 2011).

The binary parameters b_{ja} are equal to 1, if container path $j \in \Omega_k, k \in K$ uses arc $a \in A^S$ and 0 otherwise. Parameters U^e and N^e denote the capacity and the available number of vessels of a vessel class $e \in E$, respectively. The cost parameter c_a^e represents the cost of vessel class $e \in E$ sailing arc $a \in A^S$. It includes the vessel time charter rate proportional to the duration of the sailing, fuel cost for sailing and idling proportional to the sailing speed and idling time at port, port call fees, and canal transit fees (if applicable). The cost parameter c_j^k denotes the unit net cost of transporting a container of demand $k \in K$ along path $j \in \Omega_k$. This net cost is computed as the unit loading cost at origin port O_k , the unit unloading cost at destination port D_k , the transshipment cost along arcs $a \in A^T$ minus the unit revenue z_k . Parameter q_k denotes the number of containers associated with demand $k \in K$ and p^k represents a demand rejection

penalty. Note that adding such a penalty in the objective function is equivalent to increasing the revenue of a demand plus adding a constant term to the objective function value.

The LSNDSP can be formulated as the following MIP model.

$$\min \quad \sum_{a \in A^S} \sum_{e \in E_a} c_a^e y_a^e + \sum_{k \in K} \sum_{j \in \Omega_k} c_j^k x_j^k + \sum_{k \in K} \left(q^k - \sum_{j \in \Omega_k} x_j^k \right) p^k \quad (1)$$

$$\text{s.t.} \quad \sum_{j \in \Omega_k} x_j^k \leq q^k, \quad \forall k \in K \quad [\alpha] \quad (2)$$

$$\sum_{e \in E} \sum_{a \in A_e^+(v)} y_a^e \leq 1 \quad \forall v \in V \quad [\beta] \quad (3)$$

$$\sum_{a \in A_e^+(v)} y_a^e - \sum_{a \in A_e^-(v)} y_a^e = 0 \quad \forall e \in E, v \in V^e, \quad [\gamma] \quad (4)$$

$$\sum_{k \in K_a} \sum_{j \in \Omega_k} b_{ja} x_j^k - \sum_{e \in E_a} U^e y_a^e \leq 0 \quad \forall a \in A^S \quad [\lambda] \quad (5)$$

$$\sum_{a \in A^e} t_a y_a^e \leq 168 N^e \quad \forall e \in E \quad [\pi] \quad (6)$$

$$x_j^k \geq 0 \quad \forall k \in K, j \in \Omega_k \quad (7)$$

$$y_a^e \in \{0, 1\} \quad \forall a \in A^S, e \in E_a. \quad (8)$$

The objective function (1) minimizes the total net cost that is computed as the sum of the service costs and the cargo-related costs (unloading, loading, and transshipment costs plus a rejection penalty) minus the total revenue for the transported demands. Constraints (2) limit the number of containers that can be transported for each demand. Constraints (3) ensure that no two services can use the same vertex. It is a technical constraint that is required to capture transshipment costs correctly. A practical implication of that constraint under a discretized time horizon is, however, that only one service is allowed to leave a port every h hours. The coarser the time discretization h is, the more prohibitive is the constraint from a practical point of view. The implications can be mitigated by adding multiple vertices for a given port and time, possibly representing different terminals or even berths. Constraints (4) are flow conservation constraints for the service variables. For every vessel class they require that the flow into a vertex be equal to the flow out of this vertex, i.e. if a service arrives at a port at a particular time, it has to leave it again after the port stay time. Constraints (5) link the service variables to the cargo paths. They ensure that if cargo is transported from one port to another at a particular departure time and with a particular speed (sailing time), then a vessel is assigned to this sailing with a sufficient capacity. Fleet availability is enforced by constraints (6). Finally, nonnegativity and binary requirements (7)–(8) restrict the domains of the variables.

The following trivial valid inequalities, known as *strong linking inequalities* (see e.g. Frangioni and Gendron, 2009), can be used to tighten the linear programming (LP) relaxation of (1)–(8):

$$\sum_{j \in \Omega_k} b_{ja} x_j^k - \sum_{e \in E_a} \min \{q^k, U^e\} y_a^e \leq 0, \quad \forall k \in K, a \in A^S \quad [\phi] \quad (9)$$

For practical applications the total number of valid inequalities (9), $O(|A^S| \cdot |K|)$, is usually too large to consider all of them explicitly.

If the binary restrictions on variables y_a^e are relaxed, the model becomes a linear program. Below we will use α , β , γ , λ , π and ϕ to denote the dual variable vectors corresponding to constraints (2)–(6) and valid inequalities (9) of the LP relaxation of (1)–(9).

Model (1)–(8) is a variation of a service network design model (see e.g. Crainic, 2000). More specifically, the LSNDSP is a cyclic, scheduled network design problem and a generalization of the problem described by Andersen et al. (2011) or Crainic et al. (2016). Below we discuss some of the differences. In Andersen et al. (2011) and Crainic et al. (2016), the fleet of vehicles is assumed to be homogeneous and required to return to their origin within the schedule length (one week); in the LSNDSP, different vessel classes exist and the duration of a liner shipping service is only limited by the available number of vessels of a particular vessel class. Andersen et al. (2011) do not consider transshipments of commodities; Crainic et al. (2016) allow for transshipments, but no cost is imposed. Although it is a minor difference at first glance, a positive cost for transshipments makes the problem more difficult, because it generally increases the LP gap for service network design instances; transshipments occur less often when solving the LP relaxation, as fractions of services can be used to offer direct transportation for any demand and therefore, higher transshipment costs will increase the upper bound more than they will lift the lower bound. In many service network design applications, including the one by Andersen et al. (2011), all or some of the origin-destination demands *have* to be satisfied fully or partially. The resulting lower bounds on the transported demand reduce the solution space and allow to use additional sets of valid inequalities, as for example *cutset* inequalities (Magnanti et al., 1995). On the other hand, as Andersen et al. (2011) point out, lower bounds on transported demand may make it more difficult to find a feasible solution. Last but not least, origin-destination demands are only associated with a port in case of the LSNDSP, whereas in Andersen et al. (2011) they are additionally required to be served at a particular time. Again it results in a reduction of the solution space, but may increase the difficulty of finding a feasible solution.

The simpler ULSNDP is already strongly $\mathcal{NP}$ -hard (Brouer et al., 2014a) and the solution space of the LSNDSP is much larger for an equivalent problem instance; a single solution to the ULSNDP of n services corresponds to $(168/h)^n$ different solutions to the LSNDSP and vice

versa. A proof of the $\mathcal{NP}$ -hardness of the LSNDSP can be found in Appendix A.

4 A column-and-row generation matheuristic

To solve model (1)–(9), we propose a column generation matheuristic. The general algorithm design is motivated by the following observations: First, the linear relaxation of capacitated network design problems is known to be notoriously weak and no efficient exact methods exist for large instances of the general problem. Second, heuristics that do not make use of mathematical programming techniques are usually also not very successful for capacitated network design problems, as Crainic (2000) concludes.

The method we propose makes use of the efficiency of heuristics and uses LP techniques to guide the search. Algorithm 1 gives a high-level description of the column generation matheuristic (LSNDSP-CGM). Before explaining each step of the algorithm in detail, we introduce some additional notation. We use χ and X to denote a single feasible integer solution and a set of feasible integer solutions of model (1)–(9), respectively. A feasible integer solution χ is defined by a set of services S_χ and a set of cargo paths Ω_χ . A service $s \in S_\chi$ is defined by a vessel class $e(s)$ and a set of arcs A_s^S that represents the route and schedule operated by service s .

Let $M(A', E', S', \Omega', \Lambda', \Phi')$ denote a restricted version of model (1)–(9) that is dynamically updated during the execution of method LSNDSP-CGM. Set A' denotes the subset of arcs that are considered in the model, i.e. all arcs $a \in A \setminus A'$ are not modeled and their use is therefore

Algorithm 1 LSNDSP-CGM

Require: Initial (possibly empty) solution χ defined by cargo paths Ω_χ and services S_χ

Require: Parameters $T^{\max}$, $N_{\text{as}}^{\min}$, $I_{\text{crg}}^{\max}$, $N_{\text{crg}}^{\max}$, F_{MIP} , F_{Pol}

```

1: Initialize  $\Omega' \leftarrow \Omega_\chi$ ,  $\Lambda' \leftarrow \Lambda(\Omega')$ ,  $\Phi' \leftarrow \emptyset$ ,  $E' \leftarrow \emptyset$ ,  $S' \leftarrow S_\chi$ ,  $X \leftarrow \emptyset$ 
2: Set best known solution  $\chi^* \leftarrow \chi$ 
3: while (runtime <  $T^{\max}$ ) do
4:    $A' \leftarrow \text{SELECTALLOWEDARCS}(X \cup \{\chi^*\})$  ▷ Section 4.1
5:    $E', S' \leftarrow \text{FIXPARTOFNETWORK}(\chi^*)$  ▷ Section 4.2
6:   Build model  $M = M(\Omega', \Lambda', \Phi', A', E', S')$ 
7:    $\Omega', \Lambda', \Phi' \leftarrow \text{COLROWGEN}(M)$  ▷ Section 4.3
8:    $X \leftarrow \text{GENFEASIBLESOLS}(M)$  ▷ Section 4.4
9:   for  $\chi \in X$  do
10:     $\chi \leftarrow \text{CARGOALLOCATION}(\chi)$  ▷ Section 4.5
11:   end for
12:    $\chi' = \arg \min_{\chi \in X} z(\chi)$ 
13:   if  $z(\chi') < z(\chi^*)$  then
14:      $\chi^* \leftarrow \chi'$ 
15:     Delete all cargo paths:  $\Omega' \leftarrow \emptyset$ 
16:   end if
17: end while

```

not possible. The arc selection procedure is described in Section 4.1. Sets $E' \subset E$ and $S' \subseteq S$ represent subsets of fixed vessel classes and fixed services, respectively, that define variable fixing constraints. The fixing procedure and resulting variable fixing constraints are described in Section 4.2. Ω' , Λ' and Φ' denote the subsets of cargo path variables, arc capacity constraints (5) and strong linking inequalities (9), respectively, that are part of the model. For ease of reading we use the shorter notation M as equivalent to $M(A', E', S', \Omega', \Lambda', \Phi')$.

Method LSNDSP-CGM requires an initial solution, which, however, can also be an empty liner shipping network, and a set of input parameters. The algorithm starts with initializing the sets that define model $M(A', E', S', \Omega', \Lambda', \Phi')$ (step 1). Sets A' , E' and S' are initially empty and are updated at the beginning of each improvement iteration. The initial set of cargo paths Ω' contains all cargo paths of the initial solution, if any. The initial set of arc capacity constraints Λ' depends on the initial set of cargo paths and contains constraints for all arcs that are used by at least one cargo path. Formally, $\Lambda(\Omega')$ contains arc capacity constraints for the set of arcs $\{a \in A^S : \sum_{j \in \Omega'} b_{ja} > 0\}$. Set Φ' is initially empty. Note that any update of the model-defining sets implicitly updates the model as well.

The core of the algorithm is an iterative destroy-and-(re-)build mechanism described by steps 3 to 17. We provide a high-level overview of the algorithm core first and discuss its key components in detail in subsequent subsections.

At the beginning of each iteration, the solution space is reduced by only selecting a subset of arcs for reoptimization (step 4) and by partially fixing the current network (step 5), if any. Non-selected arcs are excluded and cannot be part of a solution generated during an iteration. Fixed services cannot be modified and fixed vessel classes cannot be used to design new services during an iteration. The arc selection procedure and the fixing procedure are discussed in detail in Sections 4.1 and 4.2, respectively. The main rationale behind both the selection of a subset of arcs and the partial fixing of the current network is to keep the optimization problem tractable.

The column and row generation (CRG) phase is a crucial component of the LSNDSP-CGM. The CRG procedure is based on the LP relaxation of model M , denoted $LP(M)$, and aims at generating new cargo paths (step 7). We summarize the CRG algorithm in Section 4.3 and discuss the pricing problems together with an efficient label setting algorithm in Section 4.3.1. In Section 4.3.2 we present an effective dual penalization procedure used by the CRG algorithm. The dual penalties play a key role during the CRG phase, particularly for the pricing of new cargo paths.

After the generation of cargo paths, the method generates multiple feasible integer solutions. Emphasis lies on the number and diversification of feasible networks rather than on finding the best network for the given subset of cargo paths Ω' . The generation of feasible solutions consists

of two steps: First, a MIP solver is run and a set of feasible solutions X is collected. Second, the MIP polishing heuristic of Rothberg (2007) is run to enlarge and diversify the set X (step 8). For details see Section 4.4.

Each single feasible solution $\chi \in X$ previously generated is based on the subset Ω' of cargo paths. In steps 9 to 11 the algorithm solves the Cargo Allocation Problem (CAP) for each feasible solution in order to determine the optimal cargo allocation for the network corresponding to each solution $\chi \in X$. The CAP is described in Section 4.5. The resulting set of solutions X represents a set of networks and each network's optimal cargo allocation.

If the best solution χ' among all solutions X constitutes a new global best solution, all cargo paths are removed from model M (step 15).

4.1 Selection of allowed arcs

Each iteration of algorithm LSNDSP-CGM starts with selecting a subset of arcs $A' \subseteq A$ that are allowed to be used during the current iteration. Any solution to the LSNDSP typically uses only a small fraction of all arcs and the arc selection procedure aims at reducing the problem size by selecting arcs that are likely to be part of a solution. Instead of explicitly selecting arcs, algorithm LSNDSP-CGM selects sailing legs to be allowed during an iteration. Recall that a sailing leg $l \in L$ is defined by an origin port and a destination port, and for each sailing leg multiple sailing arcs $a \in A_l^S$ exist, representing different departure times and sailing speeds. If a sailing leg $l \in L$ is selected, all corresponding arcs $a \in A_l^S$ are allowed to be used during an iteration; otherwise, they are forbidden. Transshipment arcs are never forbidden.

With each sailing leg we associate a score in the range $[0.1, 1.0]$. The score serves as an indicator for the likelihood of a sailing leg to be part of a final solution. The minimum number of sailing legs $l \in L$ to be selected at each iteration is set to the number of legs $|L_{\chi^*}|$ used in the current global best solution χ^* multiplied with a leg selection multiplier $N_{\text{leg}}^{\min}$. By definition, all legs with a score of 1.0 are included. The remaining sailing legs are randomly drawn and selected with a probability equal to their score, until the minimum number of legs to be selected is reached.

The initial score is 1.0 for all sailing legs and it is updated based on the solutions $\chi \in X \cup \{\chi^*\}$ of the last improvement iteration as follows: Let L^* denote the set of sailing legs that are part of at least one solution $\chi \in X \cup \{\chi^*\}$; all sailing legs $l \in L^*$ are assigned a score of 1.0, whereas the score of all other sailing legs is reduced by 0.1. Sailing arcs that represent the current best network are thus never forbidden. The score of a sailing leg cannot be lower than 0.1 in order to guarantee that any sailing leg has a chance to be (re-)considered in any iteration. To avoid any initial bias, no arc selection is performed during the first $|E|$ iterations but all arcs are

considered.

4.2 Partial fixing of network

At each iteration of algorithm LSNDSP-CGM, a part of the network of the current best solution is fixed. Vessel classes and services may be fixed for both practical and algorithmic reasons. Practical reasons will be given by the particular decision problem that is addressed. The decision maker may, for example, only want to re-design feeder services or introduce new services in a particular region without modifying the existing network.

From an algorithmic point of view, fixing vessel classes and services trivially reduces the complexity of the optimization problem at each iteration. For the particular optimization problem, which is a variation of a capacitated network design problem, it also helps to generate cargo paths of higher quality during the column-and-row-generation phase. We will elaborate this argument further in Section 4.3.

At each iteration, algorithm LSNDSP-CGM fixes all but one vessel classes, i.e. $|E'| = |E| - 1$. The set $E \setminus E'$, containing a single non-fixed vessel class, determines the vessel class for which new services can be generated during an iteration. The selection is random, except for the first $|E|$ iterations; in the first iteration of LSNDSP-CGM all but the largest vessel class are fixed, in the second iteration all but the second largest vessel class are fixed and so on. This deterministic fixing rule during the first $|E|$ iterations aims at preventing small vessels to be initially deployed on the trades with the largest volumes, particularly when the algorithm is used to build a network from scratch. The set of fixed services $S' \subseteq S_{\chi^*}$ is chosen to contain all services of all fixed vessel classes. Consequently, all services of the non-fixed vessel class are destroyed and rebuilt during an iteration of LSNDSP-CGM.

Let $e(s)$ denote the vessel class that is deployed on service $s \in S$. The variable fixing constraints resulting from the choice of S' are

$$y_a^{e(s)} = 1, \quad \forall a \in A_s^S, s \in S' \quad (10)$$

For fixed vessel classes $e \in E'$, all corresponding decision variables y_a^e are fixed to zero if not part of a fixed service $s \in S'$.

$$y_a^{e(s)} = 0, \quad \forall e \in E', a \in \left(A_e^S \setminus \bigcup_{s \in S'} A_s^S \right) \quad (11)$$

We have tested various other strategies for both fixing vessel classes and services, including less aggressive strategies that allow services to be generated for more than one vessel class in each iteration. However, none of these strategies outperformed the simple one described above.

4.3 Column and row generation

During the CRG phase the integrality constraints (8) are relaxed and the resulting model becomes a linear program. We call the relaxed version of model $M(A', E', S', \Omega', \Lambda', \Phi')$ the *restricted master problem* (RMP). The CRG phase aims at generating good cargo paths, rather than solving the RMP to optimality; i.e. cargo paths that are likely to be part of a feasible *integer* solution.

This motivates the following CRG algorithm design (see Algorithm 2): At each CRG iteration we first solve the RMP to obtain updated dual variable values (step 3). A set Ω'' of cargo paths is generated by solving pricing problems (step 5). The pricing problems and a label setting algorithm to solve them are described in detail in Section 4.3.1. The $N_{\text{crg}}^{\max}$ most negative reduced cost paths per demand are added to the current set of columns Ω' (step 6). Based on the newly added cargo paths, the current set of arc capacity constraints Λ' is updated by adding missing constraints (5) for all arcs that are used by at least one cargo path $j \in \Omega''$ (step 7). The updated RMP is solved (step 8) and, if activated, all violated strong linking inequalities are identified and added to the RMP (step 9).

Besides quickly increasing the number of constraints in the RMP, the strategy of adding all violated strong linking inequalities also comes at the expense of an extra call to the LP solver at each CRG iteration and increases the number of pricing problems to solve (discussed in Section 4.3.1). We will compare the algorithm performance with and without adding violated strong linking inequalities in Section 5.2.

At each call of the CRG procedure at most $I_{\text{crg}}^{\max}$ column and row generation iterations are performed.

Algorithm 2

```

1: procedure COLROWGEN( $M$ )
2:   for  $i \in \{1, \dots, I_{\text{crg}}^{\max}\}$  do
3:     Solve RMP
4:      $\tilde{\Lambda} \leftarrow \text{UPDATEDUALPENALTIES}(E')$  ▷ Section 4.3.2
5:     Generate negative reduced cost cargo paths  $\Omega''$ , at most  $N_{\text{crg}}^{\max}$  ▷ Section 4.3.1
       per demand
6:     Add generated cargo paths  $\Omega''$  to RMP:  $\Omega' \leftarrow \Omega' \cup \Omega''$ 
7:     Add arc capacity constraints:  $\Lambda' \leftarrow \Lambda' \cup \Lambda(\Omega'')$ 
8:     Solve RMP
9:     Identify and add violated strong linking inequalities  $\Phi''$ :  $\Phi' \leftarrow \Phi' \cup \Phi''$ 
10:   end for
11: end procedure

```

4.3.1 A label setting algorithm for the cargo path pricing problems

If strong linking inequalities (9) are used in model M , there is one pricing problem for each demand $k \in K$. For a demand k , the pricing problem can be formulated as a resource constrained shortest path problem defined on a graph $\bar{G}_k = (\bar{V}_k, \bar{A}_k)$. Vertex set $\bar{V}_k = V_k \cup \{o_k, d_k\}$, where o_k and d_k are the source and sink vertices. Arc set $\bar{A}_k = A_k \cup A_k^O \cup A_k^D$, where A_k^O contains arcs linking the source vertex o_k to every vertex in V_k associated with port O_k , and A_k^D contains arcs linking every vertex in V_k associated with port D_k to the sink vertex d_k . With each arc $a \in \bar{A}_k$, we associate a cost c_a , a duration t_a and a reduced cost $\bar{c}_a$. We define cost c_a of an arc $a \in \bar{A}_k$ such that it equals the loading and unloading cost of a container on arcs $a \in A_k^O$ and $a \in A_k^D$, respectively, and the transshipment cost of a container on arcs $A_k \cap A^T$. For sailing arcs, c_a is zero. The reduced cost of an arc $a \in \bar{A}_k$ depends on the actual arc cost c_a , non-positive dual variable vectors λ and ϕ and a non-negative dual penalty vector $\tilde{\lambda}$ that is defined over all sailing arcs (and described in detail in Section 4.3.2):

$$\bar{c}_a := -\lambda_a - \phi_{ak} + \tilde{\lambda}_a \quad \forall a \in \bar{A}_k \cap A^S \quad (12a)$$

$$\bar{c}_a := c_a \quad \forall a \in \bar{A}_k \setminus A^S \quad (12b)$$

The reduced cost of a cargo path $j \in \Omega_k$ is defined as $\bar{c}_j^k := -z_k - p_k - \alpha_k + \sum_{a \in \bar{A}_k} b_{ja} \bar{c}_a$. Note that all reduced arc costs $\bar{c}_a$ are non-negative.

The duration t_a associated with each arc is defined as follows: For any sailing or transshipment arc $a \in \bar{A}_k \cap A$ the duration equals the sailing and transshipment time, respectively, as defined for graph G_k . All arcs $a \in A_k^O$ and $a \in A_k^D$ have an associated duration of $t_a = 0$.

The pricing problem consists of finding in $\bar{G}_k$ a negative reduced cost shortest path between o_k and d_k that does not exceed the maximum transit time T_k . A labeling algorithm can be used to solve this problem (see e.g. Irnich and Desaulniers, 2005). We present a label setting algorithm similar to the one described by Karsten et al. (2015). Different to Karsten et al. (2015), we cannot solve one aggregate subproblem per origin port, because the reduced cost of arcs may differ for different demands $k \in K$ due to the dual variable values ϕ_{ak} (see definition (12a)). Below we describe the label setting algorithm for solving one pricing problem per demand.

A label has two components (C, T) . The C component stores the reduced cost of the path associated with the label and the T component represents the cumulative transit time along this path. Resource windows $[0, \bar{C}_{kv}]$ and $[0, \bar{T}_{kv}]$ are associated with each vertex $v \in \bar{V}_k$ in order to impose bounds on the cost and time components of each label. Different to Karsten et al. (2015) we do not leave the cost component unconstrained and use tighter resource windows for the time component. A valid upper bound on the cost component of a label can be derived from the requirement that the reduced cost has to be negative; it follows from $-z_k - p_k - \alpha_k +$

$\sum_{a \in \bar{A}_k} b_{ja} \bar{c}_a < 0$ that the cost of a label cannot exceed $z_k + p_k + \alpha_k$ and thus $\bar{C}_{kv} := z_k + p_k + \alpha_k$ is valid for any vertex v (recall that $\bar{c}_a \geq 0$ for all arcs $a \in \bar{A}_k$). Similarly, the duration of a label cannot exceed the demand's maximum transit time and thus $\bar{T}_{kv} := T_k$ is valid for any vertex v . The upper bounds $\bar{C}_{kv}$ and $\bar{T}_{kv}$ are the tightest possible upper bounds for those vertices $v \in \bar{V}_k$ that represent the destination port of demand $k \in K$. Tighter upper bounds $\bar{C}_{ku}$ and $\bar{T}_{ku}$ can be derived recursively for vertices $u \in \bar{V}_k$ by finding the tightest upper bounds that satisfy $\bar{C}_{ku} + \bar{c}_a \geq \bar{C}_{kv}$ and $\bar{T}_{ku} + t_a \geq \bar{T}_{kv}$ for all $a = (u, v) \in A$, respectively. We only apply the recursive tightening of upper bounds for the time component, because these bounds have to be calculated only once at initialization and, unlike the upper bounds on the cost component, do not have to be updated.

If model M does not contain any strong linking inequalities (9), the arc costs in the pricing problems become independent of demand $k \in K$. The pricing problems can thus be aggregated by origin ports and solved as one-to-all resource constrained shortest path problems. The generalization of the label setting algorithm and the preprocessing techniques towards one-to-all resource constrained shortest path problems is straightforward.

Standard label extension functions and a standard dominance rule are used. Labels are treated in the order of increasing path durations. The time discretization allows us to cluster the set of unprocessed labels by discretized durations. It makes any explicit sorting of labels unnecessary and therefore saves further computational time.

The label setting algorithm combined with the proposed graph based model provides a rich cargo path pricing framework that can easily be extended. A limit on the number of transshipments, as suggested by Balakrishnan and Karsten (2017), for example, can be imposed by adding a bound on the number of transshipment arcs being used by a cargo path. Like in the large majority of previous studies, we use a constant revenue function and hard transit time limits in our computational tests, but alternative definitions of a transit time dependent revenue function (as e.g. used by Wang et al., 2013) can be incorporated straightforwardly, as detailed in Koza (2017).

4.3.2 Dual penalties

The reduced cost of any cargo path $j \in \Omega_k, k \in K$ consists of a demand dependent term and the reduced cost of arcs that constitute the path. Every path by definition contains at least one sailing arc and the reduced cost of a sailing arc $a \in A^S$, as defined by equation (12a), only depends on the dual variables λ_a and ϕ_{ak} if the dual penalty $\tilde{\lambda}_a$ is zero. Both dual variables λ_a and ϕ_{ak} are zero for all arcs that are not used by any cargo path $j \in \Omega'_k, k \in K$ in the RMP, because the corresponding constraints are relaxed or, if not relaxed, non-binding in that case.

Consequently, the reduced cost $\bar{c}_a$ of any sailing arc $a \in A^S$ is zero if $\sum_{k \in K} \sum_{j \in \Omega'_k} b_{ja} = 0$. For any given solution to the RMP, we will call such an arc *unused* if additionally no capacity is deployed on that same arc ($\sum_{e \in E} y_a^e = 0$).

Together with the fact that $\bar{c}_a$ is always non-negative, we can make the following two observations:

1. All unused sailing arcs have the same reduced cost of zero, independent of the sailing distance and speed that they represent.
2. The reduced cost of an unused sailing arc is always lower or equal to the reduced cost of an arc with any deployed capacity.

For the labeling algorithm described above it has quite counter-intuitive implications: Consider a network and corresponding graph of three ports A , B and C (omitting time for simplicity). Assume one fixed liner service that sails $A \rightarrow B \rightarrow C \rightarrow A$. If sets $\Omega'_k, k \in K$ are initially empty, both cargo paths $B \rightarrow A$ and $B \rightarrow C \rightarrow A$ have the same reduced cost of zero. As direct path $B \rightarrow A$ is of shorter duration, it will dominate the longer path $B \rightarrow C \rightarrow A$ despite the fact that an empty vessel is deployed on $B \rightarrow C \rightarrow A$ and no capacity is deployed on $B \rightarrow A$.

Intuitively, the reduced cost of path $B \rightarrow A$ is underestimated. We present a simple, but effective dual variable penalization procedure to overcome this issue. We define a set of dual penalty values $\tilde{\lambda}_a$ defined over the set of sailing arcs $a \in A^S$. The similarity in notation to the dual variables λ_a is intended and the penalty values can in fact be seen as penalties of the zero-valued λ_a dual variable values for unused arcs.

Formally, we define the dual penalties as

$$\tilde{\lambda}_a := \begin{cases} \eta \cdot \min_{e \in E_a \setminus E'} \{c_a^e / U^e\}, & \text{if } \sum_{k \in K} \sum_{j \in \Omega'_k} b_{ja} = 0 \wedge \sum_{e \in E} y_a^e = 0 \\ 0 & \text{otherwise} \end{cases} \quad \forall a \in A^S \quad (13)$$

where η is a non-negative parameter which we call the *dual penalty multiplier*. The penalties $\tilde{\lambda}_a$ only take a non-zero value, if the corresponding arc is unused at the time of updating the dual penalties and if the dual penalty multiplier η is not zero. The term c_a^e / U^e represents the marginal cost of sending one more container on a vessel of class $e \in E$ on arc $a \in A^S$. The marginal cost c_a^e / U^e is relative to the distance and sailing speed associated with arc a and to the fuel consumption function of vessel class e . The penalty is set to the lowest value of all feasible vessel classes $E \setminus E'$. The penalties are updated each time before the cargo path pricing problems are solved (see Algorithm 2).

Whereas the penalty values are defined per arc, dual penalty multiplier η is a global parameter used to control the overall impact of penalties $\tilde{\lambda}$. As the penalties may increase the reduced

cost of cargo paths, they may in fact cause column generation to stop prematurely; the dual penalty multiplier η can be used to lower the penalty values e.g. if no further negative reduced cost paths can be identified. The latter ensures that optimality of the CRG procedure is not affected.

Although the definition of the penalties was motivated by practical considerations, the described problem can also be seen in the light of dual variable stabilization theory (see e.g. du Merle et al., 1999; Ben Amor et al., 2009); the initial zero dual variable values for dual variables λ_a and ϕ_{ak} are likely to be very far from their optimal values and it may take a very large number of iterations until they stabilize. The dual penalties $\tilde{\lambda}_a$ aim at providing a much better initial estimate of dual variable values λ_a .

4.4 Obtaining feasible integer solutions

The final goal of each improvement iteration of method LSNDSP-CGM is to obtain an improved scheduled liner shipping network. The generation of feasible integer solutions based on the latest RMP is divided into two steps.

First, we use a standard MIP solver to generate feasible networks defined by integer y_a^e variable values. We impose a time limit that is defined as a factor F_{MIP} times the accumulated time used to solve the RMP during the preceding CRG phase (steps 3 and 8 of Algorithm 2). We conjecture that the accumulated time for solving the RMP during the column and row generation phase is a good indicator for the difficulty of model M . An advantage of a dynamic time limit is that it adapts to different model sizes in general as well as during the execution of method LSNDSP-CGM; the model size may vary significantly between iterations depending on the outcome of the network fixing procedure (Section 4.2). The integer solutions found during the MIP solving phase are stored in a solution pool X .

Second, we apply the MIP polishing heuristic by Rothberg (2007), aiming at diversifying and enlarging the set of feasible scheduled networks X . Based on the previously generated solutions to the MIP formulation and on the corresponding MIP search tree, the MIP polishing heuristic mutates and combines existing feasible solutions, fixes common variable values and explores a fixed number of nodes in the resulting sub-MIP tree. Even though the method is not problem specific, it implicitly considers the problem structure due to its tight interaction with the MIP search tree. The MIP polishing heuristic is a randomized algorithm and part of the CPLEX solver. Equivalent to the MIP solving phase we impose a dynamic time limit F_{Pol} as a factor of the accumulated time used to solve the RMP during the preceding CRG phase.

4.5 Cargo allocation for feasible networks

The scheduled networks generated during the MIP+polishing phase are based on the restricted set Ω' of cargo paths. For any solution $\chi \in X$ there may exist further negative reduced cost cargo paths $j \in \Omega \setminus \Omega'$ with the potential to improve a solution. In order to identify them, we solve a CAP for each of the solutions in the pool X .

For each solution $\chi \in X$ we fix the binary variables y_a^e that correspond to the services $s \in S_\chi$ to one (and all other variables to zero) and evaluate whether further negative reduced cost cargo paths exist. The resulting problem is a time constrained multi-cargo flow problem that can be solved efficiently using column generation as described in Section 4.3.

The solution χ' with the lowest objective function value among all solutions $\chi \in X$ defines the best solution of the current improvement iteration.

5 Computational results

We implemented the solution algorithm in C++ and used the Lemon graph library (Dezso et al., 2011) to model the underlying graphs. CPLEX 12.7 was used to solve the LP relaxations as well as to generate and diversify the set of solutions to the MIP formulation. All computational experiments were run on a system with a Xeon E5-2680 2.8GHz processor and 48 GB memory. The results of all computational tests, upon which the below analysis is based, are made available in the online supplement.

In Section 5.1 we provide a summary of the data instances and their corresponding LSNDSP graph and model instances for a time discretization of 12 hours. In Section 5.2 we analyze the performance of algorithm LSNDSP-CGM and show how the different algorithm components contribute to the overall performance. We present and discuss results of LSNDSP-CGM for different instances in Section 5.3. In Section 5.4 we analyze the implications of integrating scheduling into liner shipping network design and compare our results with those obtained under the simplifying assumption of constant transshipment times. We further show that our proposed method outperforms the current state-of-the-art algorithm for the classic, unscheduled liner shipping network design problem. Finally, in Section 5.5 we investigate the trade-off between a larger solution space due to a finer time discretization and the increased computational complexity of the LSNDSP.

5.1 Data instances

The LSNDSP-CGM was tested on data instances from the publicly available LINER-LIB benchmark suite (<http://www.linerlib.org>). Table 1 provides a short summary of the instances; for a detailed description we refer the reader to Brouer et al. (2014a). In order to compare the

performance of LSNDSP-CGM with results from the literature, we have used a demand rejection penalty of 1000 USD, a bunker price of 600 USD per ton and a constant port stay time of 24 hours for all ports as consistently used in previous studies (see e.g. Karsten et al., 2017b). A minimum transshipment time of 48 hours between arrival of the unloading and departure of the loading vessel is assumed. Given a fixed port stay time of 24 hours per port call and vessel, it corresponds to a minimum transshipment time of 0 hours between departure of the unloading vessel and arrival of the loading vessel. The maximum transshipment time is trivially set to the minimum transshipment time plus one week.

A time discretization of $h = 12$ hours was used for the tests and analysis in the subsequent sections. We ran preliminary tests to evaluate both finer and coarser time discretizations and found 12 hours to represent a good trade-off between solution quality and problem tractability. We evaluate the impact of a finer time discretization on the solution quality and algorithm runtime in Section 5.5. The number of arcs and vertices of the resulting time-space graphs and the theoretical size of model (1)–(8) corresponding to each instance are reported in Table 2. It is noteworthy that even the smallest instance of the LINER-LIB translates into a large service network design instance in terms of vertices and arcs.

5.2 Parameter settings and algorithm performance

In a first test setting we ran method LSNDSP-CGM in order to construct liner shipping networks from scratch. The method was run with the same parameter setting on all instances. The leg

Instance	Description	Ports	Legs	Vessel Classes	Vessels	Demands
Baltic	Baltic sea, single hub Bremerhaven	12	132	2	6	22
WAF	West Africa, single hub Algeciras	20	402	2	42	37
Mediterranean	Mediterranean, multi-hub	39	1,482	3	20	365
Pacific	Trade lane Asia and US west coast	45	2,122	4	100	722
WorldSmall	World, 47 main ports	47	3,142	6	263	1764
AsiaEurope	Trade lane Europe, Middle East, and Far East regions	114	19,217	6	176	4000

Table 1: Data instances of the LINER-LIB

Instance	Graph $G(V, A)$				Model	
	$ V $	$ A $	$ A^S $	$ A^T $	constraints	binary variables
Baltic	168	7,644	5,460	2,184	5,960	8,652
WAF	280	41,888	38,248	3,640	39,001	51,016
Mediterranean	546	108,206	101,108	7,098	103,240	165,816
Pacific	630	542,262	534,072	8,190	537,402	1,044,708
WorldSmall	658	1,516,186	1,507,632	8,554	1,513,084	4,130,112
AsiaEurope	1,596	7,341,404	7,320,656	20,748	7,332,236	16,239,216

Table 2: Graph and model properties for different data instances, based on a time discretization of 12 hours.

Instance	Time limit per run (sec.)	Iters.	Time per iter. (sec.)	Running time (%)				Solutions per iter.
				Cargo path pricing	RMP	MIP+ polishing	CAP	
Baltic	300	189	1.6	2.9	12.7	63.3	19.3	3.1
WAF	3600	627	5.7	1.7	10.5	52.8	33.3	10.6
Mediterranean	14400	316	45.6	1.9	15.2	75.7	6.7	8.3
Pacific	28800	212	135.6	5.9	13.8	68.8	11.0	12.2
WorldSmall	43200	159	271.6	13.8	11.4	56.8	17.6	7.4
AsiaEurope	129600	88	1471.3	24.0	8.5	42.5	24.7	9.5

Table 3: Algorithm run statistics. Values represent averages of 12 runs per instance.

selection multiplier was set to $N_{\text{leg}}^{\min} = 5$. The number of CRG iterations per improvement iteration was set to $I_{\text{crg}}^{\max} = 3$ and the maximum number of cargo paths generated per demand and CRG iteration was set to $N_{\text{crg}}^{\max} = 5$. The time limit for the MIP solve call during each improvement iteration was set to be equal to the accumulated time spent on solving all RMPs during the preceding CRG phase ($F_{\text{MIP}} = 1.0$) and the MIP polishing heuristic was given four times the accumulated time spent on solving all RMPs ($F_{\text{Pol}} = 4.0$). The maximum number of solutions to be stored in the solution pool during the MIP solving and polishing phase was set to 50. No time limit was imposed on the solution of the CAP, i.e. the CAP was solved to optimality for each solution in the solution pool at each improvement iteration. We used a constant dual penalty multiplier of $\eta = 2.0$ in all experiments. There is one difference in the algorithm configuration between the small instances **Baltic** and **WAF** on one side and all other, medium sized and large instances on the other side: Violated strong linking inequalities (9) were only added when solving instances **Baltic** and **WAF**.

Different time limits $T^{\max}$ on the total runtime of LSNDSP-CGM were imposed for different instances according to their size. The time limit for each instance is reported in the second column of Table 3 and represents the limit for a single run. As method LSNDSP-CGM is a randomized algorithm, we ran it 12 times for each instance. Method LSNDSP-CGM proves to work consistently well on different types and sizes of liner shipping network design problem instances.

Nevertheless, the instance size affects the number of iterations that can be performed during a given time limit as well as the absolute time spent on the different algorithm phases, as the algorithm statistics in Table 3 show. The reported values are the time limit per run (sec.), number of improvement iterations, average time per iteration (sec.), time for cargo path pricing (%), time for solving RMPs (%), time for generating integer solutions (%), time for solving CAPs (%) and the average number of solutions found per iteration. The difference of the sum of running times to 100% is attributed to the other algorithm components. On average, algorithm

Figure 4: Algorithm performance under different algorithm configurations.

LSNDSP-CGM spends the largest amount of time during the MIP+polishing phase, which (by definition of the dynamic time limits F_{MIP} and F_{Pol}) is five times as large as the time spent on solving RMPs. Despite the significant number of cargo path pricing problems solved at each CRG iteration, the accumulated solution time is almost negligible for the small and medium sized instances, but accounts for 14-24% of the total runtime in case of the two largest instances. A significant share of the running time is also spent on solving the CAPs, with the share differing between instances, though.

A question that arises is how the different algorithm components described in Section 4 contribute to the overall performance. The box plots in Figures 4a to 4f compare the objective function values of the final algorithm configuration ("base") with those of other configurations; all other configurations are identical to the base configuration except for a single component. Recall that the base configuration for instances **Baltic** and **WAF** does use strong linking inequalities, whereas the base configuration for all other instances does not. "NSL" and "SL" thus correspond to the base configuration without and with strong linking inequalities, respectively. "NDP" and "NAS" represent the base configuration without dual penalties and without arc selection, respectively. The dual penalties turn out to be crucial for the overall performance,

Figure 5: Algorithm convergence over time for the best runs of each instance. The displayed MIP and CAP objective function values correspond to the best MIP and CAP objective function value among all solutions in the solution pool of an improvement iteration.

independent of the instance type and size. The arc selection procedure also leads to significant improvements, although to a lesser extent and with less consistency over different instances. The strong linking inequalities significantly improve the algorithm performance for the two smaller instances, but worsen it for the medium sized and large instances. While they generally lead to a better performance per iteration, they heavily increase the size of the RMP and do not allow to aggregate pricing problems by origin ports, thus causing much larger running times per iteration. In case of the larger instances, the gain of solution quality due to the strong linking inequalities cannot compensate for the additional running time. For this reason we only added them in the base configuration for the two smallest instances.

Figures 5a to 5f visualize the algorithm convergence over time for the best run of each instance. For smaller instances, the search space relative to the total solution space during an improvement iteration is larger due to the smaller number of vessel classes and services. This allows the objective function value of networks found at the end of an improvement iteration to deviate much more from the current best network and explains the larger fluctuation in objective function

Instance	Obj. value	Deployed cap. (%)	Trans- ported cargo (%)	Trans- shipped cargo (%)	Avg. cap. util. (%)	Avg. Speed (kn)	Butterfly services (%)	Complex services (%)	Time limit per run (sec.)
Baltic									
Best	$-2.84 \cdot 10^5$	100.0	92.1	0.0	73.3	11.9	50.0	50.0	300
Average	$-2.64 \cdot 10^5$	100.0	92.1	0.0	72.9	11.9	52.0	44.0	300
WAF									
Best	$-5.99 \cdot 10^6$	91.6	94.4	12.9	57.4	10.4	25.0	12.5	3600
Average	$-5.78 \cdot 10^6$	95.2	96.2	18.1	57.2	10.7	26.3	42.1	3600
Mediterranean									
Best	$2.44 \cdot 10^6$	75.3	68.4	46.4	54.7	11.2	42.9	28.6	14400
Average	$2.75 \cdot 10^6$	92.3	72.4	41.6	52.8	11.2	21.8	34.6	14400
Pacific									
Best	$1.48 \cdot 10^6$	88.9	84.8	24.8	73.3	13.9	23.1	53.8	28800
Average	$2.78 \cdot 10^6$	96.3	87.0	28.5	74.0	13.8	12.8	33.9	28800
WorldSmall									
Best	$-3.80 \cdot 10^7$	71.3	80.2	62.9	68.0	14.9	26.1	34.8	43200
Average	$-3.26 \cdot 10^7$	80.5	82.2	61.8	69.0	15.1	16.3	41.1	43200
AsiaEurope									
Best	$-9.33 \cdot 10^6$	58.7	62.8	68.3	66.6	14.2	8.8	29.4	129600
Average	$-2.95 \cdot 10^6$	65.9	68.0	64.2	67.1	14.1	16.3	33.5	129600

Table 4: Key statistics for the LINER-LIB instances solved by the LSNDSP-CGM method. Best and average results among all 12 runs are reported for each instance.

values for the smaller instances compared to the larger instances.

Figure 5 also reveals the importance of solving the CAP for each of the networks found during the MIP+polishing phase; in the large majority of cases and particularly for larger instances, the step identifies additional profitable cargo paths that lead to a significant improvement of the best objective function value of the MIP+polishing phase.

5.3 Scheduled networks generated by LSNDSP-CGM

In Table 4 we report key statistics for the best found network as well as averages over all networks found for each instance. Reported values are the objective function value (in USD), the deployed capacity as percentage of the total available TEU capacity, the transported volume as percentage of the total number of containers, the transshipped volume as percentage of the transported volume, the average vessel utilization (weighted by sailing distance), the average sailing speed in knots, the number of butterfly services as percentage of the total number of services, the number of complex services as percentage of the total number of services and the runtime limit. A first conclusion that we can draw from the reported key statistics is that the instances and the resulting scheduled networks are quite heterogeneous. Whereas no cargo is transshipped in the liner shipping networks found for instance **Baltic**, on average almost two thirds of all containers are transshipped at least once in the networks for instance **AsiaEurope**, with the networks of the other instances ranging in between. The average capacity utilization

rates vary between 52.8% and 74.0% and average sailing speeds range from 10.7 knots to 15.1 knots.

Interestingly, the capacity utilization of the best networks are not necessarily higher than the average rates. Both the deployed capacity and the volumes of transported cargo in the best networks are consistently lower than or equal to the corresponding average values. The findings suggest that the challenge is not only to construct a network that offers many feasible cargo routing possibilities and sufficient capacity, but to achieve the right trade-off between deployment and operating costs on one side and revenues for transporting selected cargo on the other side.

The third to last and second to last columns of Table 4 display the share of butterfly and complex services among all services of a solution. The difference to 100% represents the share of simple services. The high share of complex liner services in the final solutions suggests that these bear large potential to improve liner shipping networks. The capability of modeling complex liner services straightforwardly is one of the distinguishing features of our proposed model for the LSNDSP.

5.4 Scheduled vs. unscheduled network design

To investigate the implications of integrating scheduling into the liner shipping network design problem, we have repeated the first set of computational experiments described in Section 5.3 under the simplifying assumption of a constant transshipment time of 48 hours for any transshipment. This was implemented by setting the duration of all transshipment arcs $a \in A^T$ to 48 hours, independent of the weekly time associated with their origin and destination vertices. As discussed in the introduction, this simplified setting resembles the large majority of problem formulations for the liner shipping network design problem in the literature and thus allows us to evaluate the performance method LSNDSP-CGM against solution approaches from the literature.

Table 5 displays the objective function values and transported cargo volumes obtained by LSNDSP-CGM for the original LSNDSP as well as for the ULSNDP with constant transshipment times. The two rightmost columns report the corresponding key performance indicators of the networks found by the current state-of-the-art solution method for the classic ULSNDP by Karsten et al. (2017b).

We make two key observations: First, the best and average objective function values obtained by method LSNDSP-CGM differ significantly between scheduled and unscheduled network design for all but the two smallest instances. Much better (i.e. lower) objective function values are obtained in case transshipment times are approximated by a constant of 48 hours. This is not surprising, because the optimal solution obtained under the assumption of constant transship-

Instance	LSNDSP-CGM				Karsten et al. (2017b)	
	exact transshipment time		48h transshipment time		48h transshipment time	
	Objective value	Transported cargo (%)	Objective value	Transported cargo (%)	Objective value	Transported cargo (%)
Baltic						
Best	$-2.84 \cdot 10^5$	92.1	$-2.84 \cdot 10^5$	92.1	$-0.05 \cdot 10^5$	87.9
Average	$-2.64 \cdot 10^5$	92.1	$-1.59 \cdot 10^5$	90.5	$1.74 \cdot 10^5$	85.1
WAF						
Best	$-5.99 \cdot 10^6$	96.8	$-6.00 \cdot 10^6$	96.7	$-5.48 \cdot 10^6$	97.6
Average	$-5.78 \cdot 10^6$	96.2	$-5.78 \cdot 10^6$	96.2	$-4.89 \cdot 10^6$	91.7
Mediterranean						
Best	$2.44 \cdot 10^6$	76.6	$2.08 \cdot 10^6$	81.8	$2.19 \cdot 10^6$	83.8
Average	$2.75 \cdot 10^6$	72.4	$2.37 \cdot 10^6$	76.4	$2.65 \cdot 10^6$	79.8
Pacific						
Best	$1.48 \cdot 10^6$	88.9	$-0.94 \cdot 10^6$	91.1	$1.13 \cdot 10^6$	90.3
Average	$2.78 \cdot 10^6$	87.0	$-0.1 \cdot 10^6$	90.4	$3.44 \cdot 10^6$	89.5
WorldSmall						
Best	$-3.80 \cdot 10^7$	83.0	$-5.32 \cdot 10^7$	88.5	$-4.05 \cdot 10^7$	89.1
Average	$-3.26 \cdot 10^7$	82.2	$-4.57 \cdot 10^7$	86.4	$-3.47 \cdot 10^7$	88.1
AsiaEurope						
Best	$-0.93 \cdot 10^7$	72.6	$-2.07 \cdot 10^7$	81.4	$-1.88 \cdot 10^7$	85.6
Average	$-0.30 \cdot 10^7$	68.0	$-1.39 \cdot 10^7$	76.6	$-1.52 \cdot 10^7$	84.9

Table 5: Best and average objective function values (in USD) and transported cargo (%) per instance, comparing results obtained by algorithm LSNDSP-CGM under exact and approximated (48h) transshipment times with results from Karsten et al. (2017b)

ment times of 48 hours in fact represents a lower bound for the optimal solution of the LSNDSP. We will discuss the implications of neglecting scheduling decisions in more detail below.

Second, we observe that for the ULSNDP, method LSNDSP-CGM consistently finds better solutions than the state-of-the-art algorithm from the literature for all addressed instances. The average performance is significantly better for all instances but **AsiaEurope**. It is, however, fair to note that our algorithm requires more time per instance; for the four largest instances, our runtime limits correspond to 12, 8, 4 and 9 times the runtime limits imposed by Karsten et al. (2017b); for the smaller instances, runtime limits are not critical but method LSNDSP-CGM still performs better. Figures 5a to 5f show how the objective function values of the best runs converge over the algorithm runtime. The comparison between results for the LSNDSP and the classic ULSNDP suggests that in case service level requirements such as competitive cargo transit times are relevant, decision makers should address the integrated LSNDSP rather than the classic ULSNDP.

Table 6 compares different solution statistics for the LSNDSP against the solutions obtained for the ULSNDP, i.e. under the simplifying assumption of constant transshipment times. For the two smaller instances, none of the reported statistics differ between the LSNDSP and the ULSNDP. For the **Baltic** instance, transshipments almost do not occur. For the **WAF** instance

Instance	Deployed cap. (%)		Transported volume (%)		Transported volume (FEU)				Capacity util. (%)		Speed (kn)	
					direct		transshipped					
	exact	approx	exact	approx	exact	approx	exact	approx	exact	approx	exact	approx
Baltic	100.0	100.0	92.1	90.5	4516	4326	0	110	72.9	74.8	11.9	11.9
WAF	95.2	96.8	96.2	96.2	6730	6528	1485	1692	57.2	56.7	10.7	10.6
Mediterranean	92.3	88.7	72.4	76.4	3190	2810	2271	2954	52.8	61.5	11.2	11.2
Pacific	96.3	96.3	87.0	90.4	27469	25517	10962	14443	74.0	75.8	13.8	13.7
WorldSmall	80.5	80.1	82.2	86.4	39923	35703	65154	75175	69.0	74.0	15.1	15.0
AsiaEurope	65.9	73.0	68.0	76.6	18707	16326	33685	42628	67.1	68.6	14.1	13.9

Table 6: Comparison of key statistics of scheduled (exact) vs. unscheduled (approx.) network design. Reported values represent averages over 12 runs.

the transit time limits of demands appear to be loose enough to not matter, as almost all (95%) of the rejected demand is rejected because the origin or destination port are not served by any service, and not because of transit time restrictions.

The picture looks quite different for the four larger instances. A significantly larger amount of containers is transported and a significantly higher vessel capacity utilization rate is achieved under the simplifying constant transshipment time assumption. While the increased amount of transshipped containers (+15-31%) under the constant transshipment time assumption is in line with what one would expect, we also observe that the amount of directly transported containers is *lower* under the constant transshipment time assumption. It appears that the artificial fixing of transshipment time to 48 hours actually allows containers to use routes that include a transshipment, while these routes would violate transit time limits otherwise. Indeed, the higher utilization rates may partly result from an increased absolute amount of transported containers and partly from a higher level of consolidation, because more transportation time is available per origin-destination demand due to underestimated transshipment times. The results reflect a common trade-off: a higher level of consolidation implies less direct routes and therefore higher cargo transit times and vice versa.

5.5 Finer time discretization vs. computational complexity

All results presented above are based on a time discretization of $h = 12$ hours. Whereas the resulting model is sufficient for strategic liner shipping network design, a finer time discretization may be required at an operational planning level. At this level, additional berthing slot availability information will become available to the network planners and should be considered in the optimization model. Many terminals operate eight-hour or four-hour work shifts, and as terminals commonly charge for whole shifts, cargo carriers generally plan to arrive at the beginning of a shift. Thus, at least an eight-hour time discretization may be required to adequately reflect work shift patterns at ports. A finer time discretization, however, results in a

Instance	Graph $G(V, A)$				Model	
	$ V $	$ A $	$ A^S $	$ A^T $	constraints	binary vars
Time discretization $h=12$						
Baltic	168	7,812	5,460	2,184	5,960	8,652
WAF	280	42,168	38,248	3,640	39,001	51,016
Mediterranean	546	108,206	101,108	7,098	103,240	165,816
Pacific	630	542,262	534,072	8,190	537,402	1,044,708
Time discretization $h=8$						
Baltic	252	16,296	11,004	5,292	11,742	17,556
WAF	420	91,350	82,530	8,820	83,640	109,872
Mediterranean	819	229,425	212,226	17,199	215,240	346,794
Pacific	945	1,199,709	1,179,864	19,845	1,184,496	2,294,922
Time discretization $h=4$						
Baltic	504	59,472	38,304	21,168	39,756	60,480
WAF	840	348,180	312,900	35,280	315,081	416,640
Mediterranean	1,638	856,380	787,584	68,796	792,344	1,274,952
Pacific	1,890	4,710,048	4,630,668	79,380	4,639,206	8,948,016

Table 7: Graph and model properties for different data instances, for time discretizations of 12, 8 and 4 hours.

larger graph and model, and thus comes at a higher computational cost to run the optimization algorithm. Table 7 reports graph and model sizes for the first four LINER-LIB instances and time discretization parameters of $h = 12$, $h = 8$ and $h = 4$ hours, respectively. Whereas the number of vertices grows linearly with the number of discrete time steps, the number of arcs increases approximately quadratically.

We ran algorithm LSNDSP-CGM 12 times for each instance and each time discretization. The algorithm settings were identical to those reported above, except for the runtime limits: For the LSNDSP under a time discretization of $h = 8$ and $h = 4$ hours, the runtime limit was set to twice and four times the time limit under $h = 12$, respectively. As the time discretization of $h = 4$ hours is a divisor of $h = 12$ hours, we conducted an additional set of experiments denoted by $h = (12; 4)$ in which the networks found under $h = 12$ hours are used as a starting solution for the LSNDSP under $h = 4$ hours; the runtime limit for the algorithm in this case was set to three times the time limit under $h = 12$, such that the runtime limits under $h = 4$ and $h = (12; 4)$ are identical.

In Table 8 we report best and average objective function values for all instances and under the different time discretization parameter settings. For time discretizations of $h = 8$, $h = 4$ and the combined case $h = (12; 4)$ we further report the relative change in the objective function value compared to the base case $h = 12$. Under a time discretization of $h = 8$ hours, algorithm LSNDSP-CGM still performs reasonably well. Compared to the results obtained under $h = 12$, the results are slightly worse in case of instance *Mediterranean*, but significantly better in case of instance *Pacific*. In case of instance *Baltic* we observe a far less stable performance over all 12 runs, which explains the worse average performance.

	h=12	h=8 (time limit: 2x)		h=4 (time limit: 4x)		h=(12;4) (time limit: 4x)	
	Obj. val.	Obj. val.	% vs. h=12	Obj. val.	% vs. h=12	Obj. val.	% vs. h=12
Baltic							
Best	$-2.84 \cdot 10^5$	$-2.85 \cdot 10^5$	-0.4%	$-2.89 \cdot 10^5$	-1.7%	$-2.91 \cdot 10^5$	-2.4%
Average	$-2.64 \cdot 10^5$	$-1.48 \cdot 10^5$	+43.9%	$-0.09 \cdot 10^5$	+96.6%	$-2.72 \cdot 10^5$	-3.0%
WAF							
Best	$-5.99 \cdot 10^6$	$-5.93 \cdot 10^6$	+1.0%	$-5.90 \cdot 10^6$	+1.4%	$-6.03 \cdot 10^6$	-0.6%
Average	$-5.78 \cdot 10^6$	$-5.77 \cdot 10^6$	+0.1%	$-5.78 \cdot 10^6$	-0.1%	$-5.82 \cdot 10^6$	-0.8%
Mediterranean							
Best	$2.44 \cdot 10^6$	$2.53 \cdot 10^6$	+3.6%	$2.79 \cdot 10^6$	+14.5%	$2.42 \cdot 10^6$	-0.7%
Average	$2.75 \cdot 10^6$	$2.97 \cdot 10^6$	+7.9%	$3.19 \cdot 10^6$	+16.1%	$2.69 \cdot 10^6$	-2.2%
Pacific							
Best	$1.48 \cdot 10^6$	$1.17 \cdot 10^6$	-21.0%	$2.45 \cdot 10^6$	+66.1%	$1.29 \cdot 10^6$	-12.8%
Average	$2.78 \cdot 10^6$	$2.07 \cdot 10^6$	-25.5%	$4.17 \cdot 10^6$	+49.9%	$2.57 \cdot 10^6$	-7.8%

Table 8: Results under different time discretization parameters.

If we run algorithm LSNDSP-CGM under a time discretization of $h = 4$ hours, starting with an empty network, the solution quality deteriorates significantly compared to the results obtained under a time discretization of $h = 12$ hours, most likely due to the combinatorial explosion of the solution space. Although the time limit of algorithm LSNDSP-CGM is four times higher, the additional time cannot compensate for the significantly increased problem size.

The combined setting of $h = (12; 4)$, on the other hand, represents a viable approach to obtain solutions of high quality under a time discretization of $h = 4$ hours. The best scheduled networks found under a time discretization of $h = 12$ are improved by 0.6%–12.8%. The setting resembles an approach that first designs a network under a coarser time discretization and later performs fine tuning under a finer time discretization.

6 Concluding remarks and future research

We proposed a graph and model formulation for the integrated liner shipping network design and scheduling problem. By integrating scheduling into the classic liner shipping network design problem we account for the fact that in practice network key performance indicators like cargo transit times depend substantially on the synchronization of liner services. Additionally, the proposed model is very rich in that it allows to incorporate details which previously have only been addressed in isolated cases.

We have analyzed the relevance of integrating scheduling into the classic liner shipping network design problem by testing our model against the prevalent assumption of constant transshipment times. The results show that if the service level matters, the assumption may underestimate both transshipment times and consequently transit times of cargo. Networks or parts of networks that are designed based on the simplifying assumption may finally not be able to meet shippers’

service level requirements. Furthermore, key figures like average vessel utilization may turn out to be significantly overestimated, as the results indicate.

The richness of the proposed model comes at the cost of an increased problem complexity. We proposed an iterative solution algorithm that combines heuristic and advanced linear programming techniques into a powerful matheuristic for constructing or improving scheduled liner shipping networks. We demonstrated the effectiveness of the approach by benchmarking it on the publicly available liner shipping network design data instances. Under equal assumptions LSNDSP-CGM finds better solutions for all addressed instances.

As a next step we plan to enrich the model to further close the gap to operational practice. One practically relevant extension is the consideration of vessel payloads for fuel consumption and speed optimization, as load factors and related fuel consumption may differ significantly between headhaul and backhaul trades. Similarly, cargo carriers can only fully exploit potential savings of an optimized schedule if it is robust against interruptions; delays in liner shipping are not an exception and often lie outside the cargo carriers' sphere of influence. Quantifying and including schedule reliability into the decision making process is thus a worthy and practically relevant research direction.

The further development of the proposed solution method represents another research avenue. We believe that different definitions of the local search space may allow to improve efficiency and scalability of the algorithm. Given the very positive performance of the proposed LSNDSP-CGM on comparatively large instances, it may be worth examining if some of the ideas can be generalized and adapted towards other variations of service network design problems.

Acknowledgements

The research was supported by The Danish Strategical Research Council and The Danish Energy Technology Development and Demonstration Program (EUDP) under the ENERPLAN and GREENSHIP projects. The authors thank Mikkel Muhldorff Sigurd and his team at Maersk Line for their insights and fruitful discussions on liner shipping network design in practice, and Berit Brouer for her comments and discussions on an earlier version of the manuscript. The authors also thank three anonymous reviewers for their comments and suggestions that helped to improve the manuscript.

A Proof of $\mathcal{NP}$ -hardness of the LSNDSP

We show that the LSNDSP is $\mathcal{NP}$ -hard in the strong sense by reduction from the asymmetric traveling salesman problem (ATSP).

Proposition 1. *The LSNDSP is $\mathcal{NP}$ -hard in the strong sense.*

Proof. Let $\bar{h} = 168$ and thus $|H| = 1$. The time space graph $G(V, A)$ contains exactly one vertex per port $p \in \{1, \dots, |P|\}$ in this case and arc set A^T is therefore empty. Let vertex set V and arc set A correspond to the vertex and arc set of the ATSP. Assume a single vessel class $e \in E$ ($|E| = 1$) with capacity $U^e \geq |P| - 1$, an infinite number of available vessels $N^e = \infty$ and let arc costs c_a^e of the single vessel class e correspond to the arc costs in the ATSP. For every port $\hat{p} \in P, \hat{p} \neq 1$ we define exactly one demand $k \in K$ with port $p = 1$ as origin and port $p = \hat{p}$ as destination. Let quantity $q^k = 1$ and penalty $p^k = \infty$ for all demands $k \in K$. By solving the LSNDSP, we obtain a minimum-cost service that visits each port (i.e. vertex) exactly once in order to serve all demands. The service corresponds to a minimum-cost Hamiltonian cycle on directed graph $G(V, A)$ and is an optimal solution to the corresponding instance of the ATSP. We can thus solve the ATSP by solving a LSNDSP. $\square$

References

- Agarwal, R. and Ergun, Ö. (2008). Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science*, 42(2):175–196.
- Akyüz, M. H. and Lee, C.-Y. (2016). Service type assignment and container routing with transit time constraints and empty container repositioning for liner shipping service networks. *Transportation Research Part B: Methodological*, 88:46 – 71.
- Álvarez, J. F. (2009). Joint routing and deployment of a fleet of container vessels. *Maritime Economics & Logistics*, 11(2):186–208.
- Andersen, J., Christiansen, M., Crainic, T. G., and Grønhaug, R. (2011). Branch and price for service network design with asset management constraints. *Transportation Science*, 45(1):33–49.
- Balakrishnan, A. and Karsten, C. V. (2017). Container shipping service selection and cargo routing with transshipment limits. *European Journal of Operational Research*, 263(2):652 – 663.
- Ben Amor, H. M., Desrosiers, J., and Frangioni, A. (2009). On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167 – 1184.
- Bernhofen, D. M., El-Sahli, Z., and Kneller, R. (2016). Estimating the effects of the container revolution on world trade. *Journal of International Economics*, 98:36 – 50.
- Brouer, B. D., Álvarez, J. F., Plum, C. E. M., Pisinger, D., and Sigurd, M. M. (2014a). A base integer programming model and benchmark suite for liner-shipping network design. *Transportation Science*, 48(2):281–312.

- Brouer, B. D., Desaulniers, G., Karsten, C. V., and Pisinger, D. (2015). A matheuristic for the liner shipping network design problem with transit time restrictions. In Corman, F., Voß, S., and Negenborn, R. R., editors, *Proceedings of the 6th International Conference on Computational Logistics, Delft, The Netherlands, September 23-25, 2015*, pages 195–208. Springer International Publishing, Cham.
- Brouer, B. D., Desaulniers, G., and Pisinger, D. (2014b). A matheuristic for the liner shipping network design problem. *Transportation Research Part E: Logistics and Transportation Review*, 72:42 – 59.
- Brouer, B. D., Pisinger, D., and Spoorendonk, S. (2011). Liner shipping cargo allocation with repositioning of empty containers. *INFOR: Information Systems and Operational Research*, 49(2):109 – 124.
- Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. (2013). Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467 – 483.
- Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272 – 288.
- Crainic, T. G., Hewitt, M., Toulouse, M., and Vu, D. M. (2016). Service network design with resource constraints. *Transportation Science*, 50(4):1380–1393.
- Dezső, B., Jüttner, A., and Kovács, P. (2011). Lemon – an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23 – 45.
- Drewry (2013). Container Market – Annual Review and Forecast. Technical report, Drewry Shipping Consultants, London.
- du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, 194(1):229 – 237.
- Frangioni, A. and Gendron, B. (2009). 0–1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics*, 157(6):1229 – 1241.
- Gelareh, S. and Pisinger, D. (2011). Fleet deployment, network design and hub location of liner shipping companies. *Transportation Research Part E: Logistics and Transportation Review*, 47(6):947 – 964.
- Guericke, S. and Tierney, K. (2015). Liner shipping cargo allocation with service levels and speed optimization. *Transportation Research Part E: Logistics and Transportation Review*, 84:40 – 60.
- Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In

- Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, pages 33–65. Springer US, Boston, MA.
- Karsten, C. V., Brouer, B. D., Desaulniers, G., and Pisinger, D. (2017a). Time constrained liner shipping network design. *Transportation Research Part E: Logistics and Transportation Review*, 105:152 – 162.
- Karsten, C. V., Brouer, B. D., and Pisinger, D. (2017b). Competitive liner shipping network design. *Computers & Operations Research*, 87:125 – 136.
- Karsten, C. V., Pisinger, D., Ropke, S., and Brouer, B. D. (2015). The time constrained multi-commodity network flow problem and its application to liner shipping network design. *Transportation Research Part E: Logistics and Transportation Review*, 76:122 – 138.
- Koza, D. F. (2017). Transit time dependent revenue functions. In Koza, D. F., editor, *Models and Methods for the Design and Support of Liner Shipping Networks (PhD thesis)*, pages 62–63. Technical University of Denmark, Kgs. Lyngby.
- Maersk Line (2017). Maersk line roundtrip shipping routes. <https://www.maerskline.com/routes/roundtrip-routes>. Online, accessed 17 October 2017.
- Magnanti, T. L., Mirchandani, P., and Vachani, R. (1995). Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157.
- Meng, Q. and Wang, S. (2011). Liner shipping service network design with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, 47(5):695 – 708.
- Meng, Q., Wang, S., Andersson, H., and Thun, K. (2014). Containership routing and scheduling in liner shipping: Overview and future research directions. *Transportation Science*, 48(2):265–280.
- Mulder, J. and Dekker, R. (2014). Methods for strategic liner shipping network design. *European Journal of Operational Research*, 235(2):367 – 377.
- Notteboom, T. (2012). Container shipping. In Talley, W. K., editor, *The Blackwell Companion to Maritime Economics*, pages 230–262. Blackwell Publishing Ltd.
- Notteboom, T. E. (2006). The time factor in liner shipping services. *Maritime Economics & Logistics*, 8(1):19–39.
- Plum, C. E., Pisinger, D., Salazar-González, J.-J., and Sigurd, M. M. (2014a). Single liner shipping service design. *Computers & Operations Research*, 45:1 – 6.

- Plum, C. E., Pisinger, D., and Sigurd, M. M. (2014b). A service flow model for the liner shipping network design problem. *European Journal of Operational Research*, 235(2):378 – 386.
- Reinhardt, L. B. and Pisinger, D. (2012). A branch and cut algorithm for the container shipping network design problem. *Flexible Services and Manufacturing Journal*, 24(3):349–374.
- Rothberg, E. (2007). An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541.
- Thun, K., Andersson, H., and Christiansen, M. (2017). Analyzing complex service structures in liner shipping network design. *Flexible Services and Manufacturing Journal*, 29(3-4):535–552.
- Tran, N. K. and Haasis, H.-D. (2015a). An empirical study of fleet expansion and growth of ship size in container liner shipping. *International Journal of Production Economics*, 159:241 – 253.
- Tran, N. K. and Haasis, H.-D. (2015b). Literature survey of network optimization in container liner shipping. *Flexible Services and Manufacturing Journal*, 27(2):139–179.
- Vernimmen, B., Dullaert, W., and Engelen, S. (2007). Schedule unreliability in liner shipping: Origins and consequences for the hinterland supply chain. *Maritime Economics & Logistics*, 9(3):193–213.
- Wang, S., Meng, Q., and Liu, Z. (2013). Containership scheduling with transit-time-sensitive container shipment demand. *Transportation Research Part B: Methodological*, 54(Supplement C):68 – 83.
- WTO (2008). World trade report 2008. Trade in a globalizing world. Technical report, World Trade Organization.