Les Cahiers du GERAD ISSN: 0711–2440

Integrated liner shipping network design and scheduling

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

G–2017–100

November 2017

La collection *Les Cahiers du GERAD* est constituée des travaux de recherche menés par nos membres. La plupart de ces documents de travail a été soumis à des revues avec comité de révision. Lorsqu'un document est accepté et publié, le pdf original est retiré si c'est nécessaire et un lien vers l'article publié est ajouté.

Citation suggérée: Koza, David Franz; Desaulniers, Guy; Ropke, Stephan (Novembre 2017). Integrated liner shipping network design and scheduling, Rapport technique, Les Cahiers du GERAD G-2017-100, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (https://www.gerad.ca/fr/papers/G-2017-100) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

The series *Les Cahiers du GERAD* consists of working papers carried out by our members. Most of these pre-prints have been submitted to peer-reviewed journals. When accepted and published, if necessary, the original pdf is removed and a link to the published article is added.

Suggested citation: Koza, David Franz; Desaulniers, Guy; Ropke, Stephan (November 2017). Integrated liner shipping network design and scheduling, Technical report, Les Cahiers du GERAD G-2017-100, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (https://www.gerad.ca/en/papers/G-2017-100) to update your reference data, if it has been published in a scientific journal.

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2017
– Bibliothèque et Archives Canada, 2017

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2017
– Library and Archives Canada, 2017

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Integrated liner shipping network design and scheduling

David Franz Koza [a]
Guy Desaulniers [b]
Stephan Ropke [a]

[a] *Department of Management Engineering, Technical University of Denmark, Danemark*

[b] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada*

dakoz@dtu.dk
guy.desaulniers@gerad.ca
ropke@dtu.dk

November 2017
Les Cahiers du GERAD
G–2017–100
Copyright © 2017 GERAD, Koza, Desaulniers, Ropke

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs. Les auteurs conservent leur droit d'auteur et leurs droits moraux sur leurs publications et les utilisateurs s'engagent à reconnaître et respecter les exigences légales associées à ces droits. Ainsi, les utilisateurs:

- Peuvent télécharger et imprimer une copie de toute publication du portail public aux fins d'étude ou de recherche privée;
- Ne peuvent pas distribuer le matériel ou l'utiliser pour une activité à but lucratif ou pour un gain commercial;
- Peuvent distribuer gratuitement l'URL identifiant la publication.

Si vous pensez que ce document enfreint le droit d'auteur, contactez-nous en fournissant des détails. Nous supprimerons immédiatement l'accès au travail et enquêterons sur votre demande.

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*. Copyright and moral rights for the publications are retained by the authors and the users must commit themselves to recognize and abide the legal requirements associated with these rights. Thus, users:

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

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.

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 45 ports and finds new best solutions to all of them, 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.

Acknowledgments: 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 would like to thank Berit Brouer for her comments and discussions on an earlier version of the manuscript.

1 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 liner shipping companies to transport containers between almost any possible pair of ports around the globe.

The question of how to (re-)design a liner shipping network is at the core of optimization problems faced by cargo carriers and involves various tradeoffs. While shippers generally request economic, fast, and ideally direct port-to-port connections, shipping companies 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 between 2000 and 2014 and the capacity of the largest vessels has quadrupled since the Panamax era in the mid-1990s (Tran and Haasis, 2015a). 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.

A large share of transshipped containers requires liner service schedules to be well synchronized, because they 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.

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 45 ports. To summarize, the contributions are:

- The liner shipping network design and scheduling problems are integrated into a single model.
- Transshipment times are modeled in an exact way and not approximated by a constant. Therefore, transit times are also exact.
- The method can efficiently handle service level requirements as e.g. transit time limits for origin-destination demands, transit time dependent revenue functions or limits on the number of transshipments.
- The model and solution method allow for any type of liner service and are not limited to simple or butterfly type rotations.
- The solution method optimizes sailing speed on each sailing leg.
- We propose 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 45 ports. We report new best solutions for all four considered LINER-LIB instances.
- The benefits of integrating scheduling with classic liner shipping network design are analyzed in detail.

The paper is organized as follows. Section 2 reviews related literature. In Section 3 we state and describe the LSNDSP. In Section 4 we first define a rich graph model (Section 4.1). Based on this model, we then state a mathematical formulation of the problem (Section 4.2) and shortly outline similarities and differences with other service network design models. In Section 5 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 6 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 7.

2 Literature review

The LSNDSP 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 of these problems in different contexts 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 studies that address liner shipping *network* design problems that explicitly consider transshipments of containers between 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.

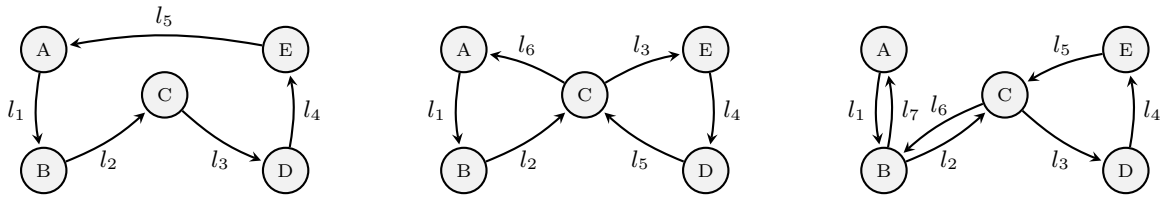
Álvarez (2009) presents a two-tier solution approach for the ULSNDP based on tabu search and column generation that is tested on a case study consisting of 120 ports. Gelareh and Pisinger (2011) propose a mixed integer programming formulation for regional liner shipping hub-and-spoke network design, 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) developed 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. (2017a) 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 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.

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. (2014) 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.



(a) Simple service **(b) Butterfly service** **(c) Complex service**

Figure 1: Illustration of different service types.

Thun et al. (2016) 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 4 we will discuss differences between the LSNDSP and existing service network design problems and models in more detail.

3 Problem statement

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. 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) 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.

A vessel class is defined by its capacity in forty-foot equivalent (FEU) units, 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 units 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. Demands of different revenues and time-sensitivities may exist for the same origin-destination pairs. Cargo can generally be transshipped between services, but every transshipment implies additional costs for cargo handling at the transshipment port. The transshipment time 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 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.

The objective 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.

4 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 4.1. The mathematical model of the LSNDSP is presented in Section 4.2.

4.1 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 steps of length $\bar{h}$ (a divisor of 168 hours), represented by the set $H = \{0, 1 \cdot \bar{h}, 2 \cdot \bar{h}, \ldots, 168 - \bar{h}\}$. 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.

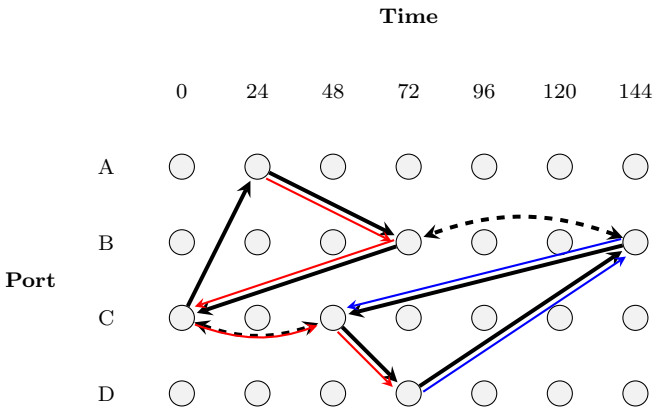


Figure 3: Illustration of a time-space graph G. Solid arcs represent sailings between ports and dashed arcs represent transshipments. The colored arcs denote examples of cargo paths; the blue path from port D to port C is direct, whereas the red path from port A to port D requires a transshipment at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

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 over a cylinder, 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 $(8 + 128 + 24) \mod 168 = 160$ and the latest weekly departure time is $(8 + 152 + 24) \mod 168 = 16$. 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 a different speed.

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, $(i_1, 32), (i_1, 40), \ldots, (i_1, 160), (i_1, 0), \ldots, (i_1, 24)$, in the order of increasing duration.

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 due to the weekly cylinder). 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. 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.

4.2 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 represented by arc $a \in A^S$ is sailed by vessel class $e \in E$ ($y_a^e = 1$) or not ($y_a^e = 0$). 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.

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

$$\text{s.t.} \quad \sum_{j\in\Omega^k} x_j^k \le q^k, \qquad \forall k \in K \qquad [\alpha] \tag{2}$$

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

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

$$\sum_{k\in K_a}\sum_{j\in\Omega^k} b_{ja} x_j^k - \sum_{e\in E_a} U^e y_a^e \le 0 \qquad \forall a \in A^S \qquad [\lambda] \tag{5}$$

$$\sum_{a\in A^e} t_a y_a^e \le 168 N^e \quad \forall e \in E \qquad [\pi] \tag{6}$$

$$x_j^k \ge 0 \qquad \forall k \in K, j \in \Omega^k \tag{7}$$

$$y_a^e \in \{0,1\} \qquad \forall a \in A^S, e \in E_a. \tag{8}$$

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 depart from the same port at the same time. 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* (Frangioni and Gendron, 2009), can be used to tighten the linear programming (LP) relaxation of (1)–(8):

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

For practical applications the total number of valid inequalities (9), $O(|A^S| \times |K|)$, 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 $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{\lambda}$, $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ 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.

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 $|H| = 1$. Time space graph $G(V, A)$ contains exactly one vertex per port $p \in \{1, \ldots, |P|\}$ 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. □

5 A column generation matheuristic

To solve model (1)–(8), 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 $M(\Omega', \Lambda', \Phi', \mathcal{T}, E', S')$ denote a restricted version of model (1)–(9) that is dynamically updated during the execution of method LSNDSP-CGM. 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. Set $\mathcal{T}$ represents tabu constraints that the method LSNDSP-CGM may add to the model. The tabu constraints are described in detail in Section 5.7. Similarly, 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 5.1. For ease of reading we use the shorter notation M as equivalent to $M(\Omega', \Lambda', \Phi', \mathcal{T}, E', S')$.

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 $M(\Omega', \Lambda', \Phi', \mathcal{T}, E', S')$ (step 1). 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 $\{a \in A^S : \sum_{j \in \Omega'} b_{ja} > 0\}$. All other sets Φ', $\mathcal{T}$, E' and S' are initially empty. Model $M(\Omega', \Lambda', \Phi', \mathcal{T}, E', S')$ is generated based on the initial sets (step 2). 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 4 to 19. 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 current network is partially fixed (step 5). Fixed services cannot be modified and fixed vessel classes cannot be used to design new services during an iteration. The main rationale behind the partial fixing of the network is to keep the optimization problem tractable. The fixing procedure is discussed in detail in Section 5.1.

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 6). We summarize the CRG algorithm in Section 5.2 and discuss the pricing problems together with an efficient label setting algorithm in Section 5.3. In Section 5.4 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 the optimal network for the given 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 7). For details see Section 5.5.

Algorithm 1 LSNDSP-CGM

Require: Initial (possibly empty) solution χ defined by cargo paths Ω_χ and services S_χ
Require: Parameters $T^{\max}$, $I^{\max}_{\text{crg}}$, $N^{\max}_{\text{crg}}$, F_{MIP}, F_{Pol}
1: Initialize $\Omega' \leftarrow \Omega_\chi$, $\Lambda' \leftarrow \Lambda(\Omega')$, $\Phi' \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$, $E' \leftarrow \emptyset$, $S' \leftarrow S_\chi$
2: Build model $M = M(\Omega', \Lambda', \Phi', \mathcal{T}, E', S')$
3: Set best known solution $\chi^* \leftarrow \chi$
4: **while** (*runtime* $< T^{\max}$) **do**
5: $E', S' \leftarrow$ FixPartOfNetwork(M, χ^*) ▷ Section 5.1
6: $\Omega', \Lambda', \Phi' \leftarrow$ ColRowGen(M) ▷ Sections 5.2 to 5.4
7: $X \leftarrow$ GenFeasibleSols(M) ▷ Section 5.5
8: **for** $\chi \in X$ **do**
9: $\chi \leftarrow$ CargoAllocation(χ) ▷ Section 5.6
10: **end for**
11: $\chi' = \arg\min_{\chi \in X} z(\chi)$
12: **if** $z(\chi') < z(\chi^*)$ **then**
13: $\chi^* \leftarrow \chi'$
14: Delete all cargo paths: $\Omega' \leftarrow \emptyset$
15: Delete all tabu constraints: $\mathcal{T} \leftarrow \emptyset$
16: **else**
17: Add tabu constraints for solution χ': $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}(\chi')$ ▷ Section 5.7
18: **end if**
19: **end while**

Each single feasible solution $\chi \in X$ previously generated is based on the subset Ω' of cargo paths. In steps 8 to 10 the algorithm solves the Cargo Allocation Problem (CAP) for each feasible solution in order to find *all* negative reduced cost cargo paths for the fixed network corresponding to each solution χ. The CAP is described in Section 5.6. 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 and tabu constraints are removed from model M (steps 14 to 15). Otherwise, tabu constraints $\mathcal{T}(\chi')$ that forbid solution χ' in subsequent iterations are added to increase diversification (step 17). The tabu constraints are described in more detail in Section 5.7.

5.1 Partial fixing of network

Each iteration of algorithm LSNDSP-CGM starts with fixing a part of the network of the current best solution. 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 5.2.

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

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

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 $s \in S'$.

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

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.

5.2 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 $M(\Omega', \Lambda', \Phi', \mathcal{T}, E', S')$ 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 5.3. 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 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 rather aggressive 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 5.3). However, preliminary experiments have shown that the gain in solution quality significantly outweighs the increased running time per CRG iteration.

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

5.3 A label setting algorithm for the cargo path pricing problems

There is a 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 $\boldsymbol{\lambda}$ and $\boldsymbol{\phi}$ and a non-negative dual penalty vector $\tilde{\boldsymbol{\lambda}}$ that is defined over all sailing arcs (and described in detail in Section 5.4):

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

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

The reduced cost of a cargo path is defined as $\bar{c}_j^k := -z_k - \alpha_k + \sum_{a \in \bar{A}_k} b_{ja} \bar{c}_a$. 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

Algorithm 2

```
1: procedure ColRowGen(M)
2:     for i ∈ {1, ..., I_crg^max} do
3:         Solve RMP
4:         λ̃ ← UpdateDualPenalties(E')                          ▷ Section 5.4
5:         Generate negative reduced cost cargo paths Ω'', at most N_crg^max   ▷ Section 5.3
           per demand
6:         Add generated cargo paths Ω'' to RMP: Ω' ← Ω' ∪ Ω''
7:         Add arc capacity constraints: Λ' ← Λ' ∪ Λ(Ω'')
8:         Solve RMP
9:         Identify and add violated strong linking inequalities Φ'': Φ' ← Φ' ∪ Φ''
10:    end for
11: end procedure
```

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 $-z_k - \alpha_k + \sum_{a \in \bar{A}_k} b_{ja} \bar{c}_a < 0$ that the cost of a label cannot exceed $z_k + \alpha_k$ and thus $\bar{C}_{kv} := z_k + \alpha_k$ 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. This preprocessing technique can be generalized towards one-to-all shortest path problems.

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.

5.4 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 ($\sum_{e \in E} y^e_a = 0$).

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

1. Two 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 $A \to B \to C \to A$. If sets $\Omega'_k, k \in K$ are initially empty, both cargo paths $B \to A$ and $B \to C \to A$ have the same reduced cost of zero. As direct path $B \to A$ is of shorter duration, it will dominate the longer path $B \to C \to A$ despite the fact that an empty vessel is deployed on $B \to C \to A$ and no capacity is deployed on $B \to A$.

Intuitively, the reduced cost of path $B \to A$ 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

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

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; 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.

5.5 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 5.1). 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 MIP solutions and 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. 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.

5.6 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 (MCF) problem that can be solved efficiently using column generation as described in Section 5.2.

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

5.7 Tabu constraints

During the MIP+polishing phase the same networks may be generated in subsequent iterations. In order to avoid it partly and improve diversification, we add a tabu constraint to model M at the end of each improvement iteration. The constraint forbids the combination of redesigned services (excluding the fixed services) of the best solution $\chi' \in X$ in subsequent improvement iterations. The constraints are of the form $\sum_{e \in E \setminus E'} \sum_{a \in \hat{A}_e} y_a^e \leq |\cup_{e \in E \setminus E'} \hat{A}_e| - 1$ with $\hat{A}_e$ denoting the arcs that are used by any service of vessel class e. I.e. the constraints forbid only the set of services that has been re-designed during an iteration. Tabu constraints in set $\mathcal{T}$ are activated and deactivated dynamically at each iteration depending on which part of the network is fixed. Otherwise a tabu constraint might be in conflict with the network fixing constraints. Whenever a new global best solution is found, the set $\mathcal{T}$ of tabu constraints is deleted.

6 Computational results

We implemented the solution algorithm in C++ and used the Lemon graph library (Dezsõ et al., 2011) to model the underlying graphs. CPLEX 12.7 was used to solve the LP relaxations as well as to generate MIP solutions. All computational experiments were run on a system with a Xeon E5-2680 2.8GHz processor and 48 GB memory.

In Section 6.1 we provide a summary of the data instances and their corresponding LSNDSP graph and model instances. We present and discuss results of the LSNDSP-CGM for different instances in Section 6.2. In Section 6.3 we analyse 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; for the latter case we present both our own experiments as well as results from a state-of-the-art algorithm for unscheduled network design from the literature. Finally, Section 6.4 provides insights about the algorithm performance for different instances.

6.1 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 fairly compare the performance of LSNDSP-CGM

Table 1: Data instances of the LINER-LIB.

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

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 was assumed. The maximum transshipment time is set to the minimum transshipment time plus one week, given the weekly service frequency.

A time discretization of 12 hours was used for the graph model, representing a good trade-off between solution quality and problem tractability. A finer time discretization implies much larger graphs, whereas a coarser time discretization can be too restrictive and result in solutions of poor quality, as preliminary tests have shown. The number of arcs and vertices of the resulting time-space graphs and the theoretical size of model (1)–(8) corresponding to each data instance are reported in Table 2. It is noteworthy that even the smallest data instance of the LINER-LIB translates into a large service network design instance in terms of vertices and arcs. In case of instance `WorldSmall` the large size of the resulting LSNDSP instance caused the LSNDSP-CGM to run out of memory. In the remainder of this section we therefore report results only for the instances `Baltic`, `WAF`, `Mediterranean` and `Pacific`.

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

Instance	Graph $G(V, A)$				Model	
	$\|V\|$	$\|A\|$	$\|A^S\|$	$\|A^T\|$	constraints	binary vars
Baltic	168	7,812	5,460	2,352	5,988	8,652
WAF	280	42,168	38,248	3,920	39,127	51,016
Mediterranean	546	108,206	101,108	7,644	103,660	165,816
Pacific	630	542,262	534,072	8,820	537,948	1,044,708
WorldSmall	658	1,516,844	1,507,632	9,212	1,514,008	4,130,112

6.2 LSNDSP-CGM results

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 number of CRG iterations per improvement iteration was set to $I^{\max}_{\text{crg}} = 3$ and the maximum number of cargo paths generated per demand and CRG iteration was set to $N^{\max}_{\text{crg}} = 5$. 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 preceeding CRG phase ($F_{\text{MIP}} = 1.0$) and the MIP polishing heuristic was given three times the accumulated time spent on solving all RMPs ($F_{\text{Pol}} = 3.0$). 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. 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 last column of Table 3.

As method LSNDSP-CGM is a randomized algorithm, we ran it 12 times for each data instance. Table 3 reports key statistics for the best run and the average of all runs for each data instance. A first conclusion that we can draw from the reported key statistics is that the data instances and the resulting scheduled networks are quite heterogeneous. Whereas basically no cargo is transshipped in the liner shipping networks found for instance `Baltic`, around half of all transported containers are transshipped at least once in the solutions for `Mediterranean`. The average capacity utilization rates vary between 53.7% and 73.7% and average sailing speeds range from 10.8 knots to 13.4 knots.

The last two columns of Table 3 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 feature of our proposed model for the LSNDSP.

Table 3: Key statistics for the LINER-LIB instances solved by the LSNDSP-CGM method. Best and average results are reported for each instance. Reported values are: objective value (in USD), deployed capacity as percentage of total available TEU capacity, transported volume as percentage of total number of containers, transshipped volume as percentage of transported volume, average vessel utilization (weighted by sailing distance), average speed in knots (weighted by sailing distance), number of butterfly services as percentage of total number of services, number of complex services as percentage of total number of services and run time limit.

Instance	Obj. val.	Deployed cap. (%)	Transp. vol. (%)	Trans-shipped vol. (%)	Avg. cap. util. (%)	Avg. Speed (kn)	Butterfly ser-vices (%)	Complex ser-vices (%)	Time limit (sec.)
Baltic									
Best	$-2.84{\cdot}10^5$	100.0	92.1	0.0	73.4	11.9	50.0	50.0	900
Average	$-2.24{\cdot}10^5$	100.0	92.0	0.9	73.7	12.3	48.1	37.0	900
WAF									
Best	$-5.92{\cdot}10^6$	94.4	96.8	17.7	57.2	10.8	20.0	40.0	3600
Average	$-5.76{\cdot}10^6$	95.5	96.3	23.7	57.3	10.8	22.9	37.1	3600
Mediterranean									
Best	$2.54{\cdot}10^6$	100.0	77.2	49.6	55.2	11.1	0.0	42.9	14400
Average	$2.73{\cdot}10^6$	96.2	73.8	43.3	53.7	10.9	25.0	33.8	14400
Pacific									
Best	$2.69{\cdot}10^6$	93.7	85.4	20.2	75.4	13.7	12.5	25.0	28800
Average	$3.71{\cdot}10^6$	98.2	84.9	23.8	72.9	13.4	14.1	29.7	28800

6.3 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 6.2 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. Furthermore, testing our solution method LSNDSP-CGM under these assumptions allows us to directly compare our method against algorithms proposed in the literature.

Table 4 displays the objective function values obtained by LSNDSP-CGM for the original LSNDSP as well as for the LSNDSP under the simplifying assumption of constant transshipment times. In the third column, the objective function values found by the current state-of-the-art solution method for the liner shipping network design problem by Karsten et al. (2017b) are reported. We make two key observations.

Table 4: Best and average objective function values (in USD) per instance, comparing results obtained by algorithm LSNDSP-CGM under exact and approximated (48h) transshipment times with results from Karsten et al. (2017b).

Instance	LSNDSP-CGM		Karsten et al. (2017b)
	exact transship. time	48h transship. time	48h transship. time
Baltic			
Best	$-2.84{\cdot}10^5$	$-2.84{\cdot}10^5$	$-0.05{\cdot}10^5$
Average	$-2.24{\cdot}10^5$	$-2.08{\cdot}10^5$	$1.74{\cdot}10^5$
WAF			
Best	$-5.92{\cdot}10^6$	$-5.90{\cdot}10^6$	$-5.48{\cdot}10^6$
Average	$-5.76{\cdot}10^6$	$-5.77{\cdot}10^6$	$-4.89{\cdot}10^6$
Mediterranean			
Best	$2.54{\cdot}10^6$	$2.11{\cdot}10^6$	$2.19{\cdot}10^6$
Average	$2.73{\cdot}10^6$	$2.33{\cdot}10^6$	$2.65{\cdot}10^6$
Pacific			
Best	$2.69{\cdot}10^6$	$-0.33{\cdot}10^6$	$1.13{\cdot}10^6$
Average	$3.71{\cdot}10^6$	$1.06{\cdot}10^6$	$3.44{\cdot}10^6$

First, the average objective function values obtained by method LSNDSP-CGM differ significantly between scheduled and unscheduled network design (p-value < 0.0001) for the two large 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 transshipment times of 48 hours in fact represents a lower bound for the optimal solution of the LSNDSP. For the two smaller instances the differences in the objective function values are not significant. We will discuss the implications of neglecting scheduling in more detail shortly.

Second, we observe that for the unscheduled network design problem, method LSNDSP-CGM consistently finds better solutions than the state-of-the-art algorithm from the literature for all addressed instances. It is, however, fair to note that our algorithm was given more time per instance (e.g. 28800 seconds vs. 3600 seconds for `Pacific` in Karsten et al., 2017b).

Table 5 compares different solution statistics for the LSNDSP against the solutions obtained under the simplifying assumption of constant transshipment times. For the two smaller instances, none of the reported statistics differ between the LSNDSP and the LSNDSP under the assumption of constant transshipment times. For the `Baltic` instance transshipments do almost not occur. For the `WAF` instance the transit time limits of demands appear to be loose enough to not matter, as almost all (97%) 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.

Table 5: Comparison of key statistics of scheduled (exact) vs. unscheduled (approx.) network design. Reported values represent averages over 12 runs. For underlined pairs of values the difference is statistically significant at the 0.01 level (based on Welch's t-test).

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	92.0	92.4	4477	4524	37	5	73.7	72.8	12.3	12.2
WAF	95.5	94.2	96.3	95.5	5406	5296	2739	2874	57.3	57.4	10.8	10.8
Mediterranean	96.2	96.8	73.8	79.3	3523	2667	2002	3400	53.7	60.6	10.9	11.0
Pacific	98.2	97.9	84.9	89.2	29080	24362	8419	14375	72.9	74.8	13.4	13.6

The picture looks quite different for the 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 (+70%) 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 take too long 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 demand due to underestimated transshipment times. The results reflect the common trade-off: a higher level of consolidation implies less direct routes and therefore higher cargo transit times and vice versa.

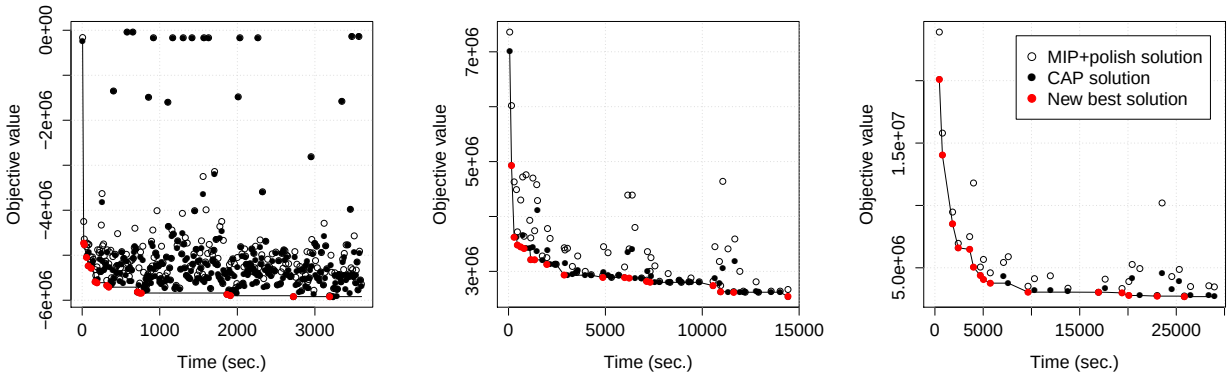
6.4 Analysis of solution algorithm performance

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 6 show. On average algorithm 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 four times as large as the time spent on solving RMPs. Despite the large number of cargo path pricing problems solved at each CRG iteration, the accumulated solution time is almost negligible for the small instances and accounts for only 10% of the total runtime even in case of instance `Pacific`.

Table 6: Algorithm run statistics. Values represent averages of 12 runs per instance. Time limit (sec.), number of improvement iterations, avg. time per iteration (sec.), time for cargo path pricing (%), time for solving RMPs (%), time for generating integer solutions (%), time for solving Cargo Allocation Problems (%) and avg. number of solutions found per iteration. The difference of the sum of running times (%) to 100% attributes to the other algorithm components.

Instance	Time limit	Iterations	Time per iter.	Running time (%)				Solutions per iter.
				Cmp pricing	RMP	MIP+ polishing	CAP	
Baltic	900	274	3.3	1.0	17.6	71.0	8.8	2.7
WAF	3600	329	11.0	0.9	14.2	58.0	25.7	10.9
Mediterranean	14400	80	179.1	5.6	17.5	70.0	6.6	12.9
Pacific	28800	37	785.5	10.2	14.1	56.8	18.7	23.6

Figures 4a to 4c 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 values for the small instance `WAF` compared to the large instances `Mediterranean` and `Pacific`.



(a) Algorithm convergence WAF

(b) Algorithm convergence Mediterranean

(c) Algorithm convergence Pacific

Figure 4: Algorithm convergence over time for the best runs of WAF, Mediterranean and Pacific. 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.

Figure 4 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. Table 7 reports the relative number of iterations with an improvement and the average improvement per iteration of the best CAP solution compared to the best MIP+polishing solution of the same iteration. The reported values are calculated over all iterations of all runs. Like indicated by Figure 4, the approach pays off the most for larger instances.

Table 7: Impact of solving a CAP for each solution found during the MIP+polishing phase. The first column reports the relative number of iterations in which the solution of the CAP was strictly better than the solution of the MIP+polishing phase. The second and third column report the absolute and relative average improvement, including the iterations without any improvement (averaged over all iterations of all runs).

Instance	Iters (%) w. improvement	Avg. improvement	Avg. improvement (%)
Baltic	49.0	51768	11.8
WAF	75.3	129897	2.5
Mediterranean	85.9	403203	9.4
Pacific	99.3	1917247	17.5

7 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 consistently 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 the method can be generalized and adapted towards other variations of service network design problems.

References

Agarwal, R. and Ergun, Ö. (2008). Ship scheduling and network design for cargo routing in liner shipping. Transportation Science, 42(2):175–196.

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 Grnhaug, R. (2011). Branch and price for service network design with asset management constraints. Transportation Science, 45(1):33–49.

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., Alvarez, 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.

Dezsõ, 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). 01 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.

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.

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., and Sigurd, M. M. (2014). 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. (2016). Analyzing complex service structures in liner shipping network design. Flexible Services and Manufacturing Journal. https://link.springer.com/article/10.1007/s10696-016-9262-6.

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.

WTO (2008). World trade report 2008. Trade in a globalizing world. Technical report, World Trade Organization.