A matheuristic for the liner shipping network design problem

Brouer, Berit Dangaard; Desaulniers, Guy; Pisinger, David

Publication date:
2014

Document Version
Peer reviewed version

Chapter 5

A matheuristic for the Liner Shipping Network Design Problem

Berit Dangaard Brouer* Guy Desaulniers†

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
blof@man.dtu.dk

† GERAD, École Polytechnique Montréal
Département de mathématiques et génie industriel,C.P. 6079, Succ. Centre-Ville,Montréal,
Québec, Canada H3C 3A7
guy.desaulniers@gerad.ca

[1]

Abstract We present a *matheuristic*, an integer programming based heuristic, for the Liner Shipping Network Design Problem. The liner Shipping Network Design Problem is to find a set of container shipping routes defining a capacitated network on which a set of demands can be transported. The cargo transports make extensive use of transshipments between services. The services have weekly frequency. The weekly frequency is an industry standard as liner shipping companies publish a set of scheduled services to their customers. The heuristic applies a greedy construction heuristic, where the liner shipping network is split into routes solving a multiple quadratic knapsack problem. The construction heuristic is combined with an improvement heuristic with a neighborhood defined as a mixed integer program. The mixed integer program optimizes the removal and insertion of several port calls on a liner shipping service. The objective function is based on evaluation functions for revenue and transshipment of cargo along with in/decrease of vessel- and operational cost for the current solution. The evaluation functions may be used by heuristics in general to evaluate changes to a network design without solving the underlying large scale multicommodity flow problem. Computational results are reported for the benchmark suite *LINER-LIB 2012* and are the first results reported for a strict weekly frequency. The heuristic minimizes the operational cost and computational results are able to find profitable transportation networks for four out of six cases (12/18 instances). The heuristic shows overall good performance and is able to find profitable solutions within a very competitive execution time. The matheuristic is also evaluated as a decision support tool, where the initial solution is an existing network and optimization is allowed for a subset of the services considering the flow in the entire network. Results are promising for this approach.

Keywords: liner shipping, matheuristic, mathematical programming, network design

ROUTES	SERVICES
West Coast of North America - Asia	68
East Coast of North America - Asia	22
North America - Northern Europe	17
North America - Mediterranean	20
Asia - North Europe	35
Asia - Mediterranean	40
North America - East Coast of South America	8
North America - West Coast of South America	16
North America - North Coast of South America	26
Europe - East Coast of South America	10
Europe - West Coast of South America	8
Europe - North Coast of South America	10
Asia - East Coast of South America	8
Asia - West Coast of South America	13
South Africa - Europe	5
South Africa - North America	2
South Africa - Asia	19
West Africa - Europe	37
West Africa - North America	3
West Africa - Asia	18
Total	385

Table 5.1: Worldwide services from WSC (2011). Notes: Services may be counted on more than one route. "Asia" includes Australia and New Zealand. Source: ComPair Data World Liner Supply Report Summary, October 1, 2010; Drewry, Container Forecast Q3 2010; Drewry, Container Forecast Q4 2010.

5.1 Introduction

Liner shipping is the mass transit system of the ocean ways with regular scheduled services of varying capacity between geographical regions. A service is a sequence of port calls sailed by a number of vessels with a designated capacity. It is common for the industry to call every port on a service every week, which is achieved by the deployed vessels sailing exactly one week apart. Liner shipping and containerized transportation of goods over sea is a key component in todays supply chains. Approximately 400 liner shipping services are operated by a vessel fleet of close to 6000 container vessels (WSC (2011)). The services are distributed on regional areas as seen in Table 5.1. The liner shipping industry transports about 60% of the goods measured by value transported internationally by sea (WSC (2011)). The significance and magnitude of the liner shipping network makes the network design an important transportation problem. The network has high fixed asset costs in terms of the container vessels deployed and hence capacity utilization and network efficiency is crucial to a competitive liner shipping operation. At the same time maritime transport is accountable for an estimated 2.7% of the worlds CO2 emissions, whereof 25% is attributable to container ships alone (WSC, 2009). Fuel cost is the largest variable cost of operating a liner shipping network (Stopford, 2009). Performing optimization on the liner shipping network can have a huge impact on the trade of liner shipping as maximizing the revenue while considering variable operational cost may ensure a better capacity utilization in the network. Improved capacity utilization will increase profit for liner shipping companies, and give competitive freight rates for global goods. In due time optimization may target reducing the speed of the container fleet to decrease the CO2 emissions from liner shipping in general as seen in the case of tramp shipping (Fagerholt et al., 2009).

The liner shipping network design problem (LSNDP) is to construct a set of non-simple cyclic services to form a capacitated network for the transport of containerized cargo. The network design maximises the revenue of container transport considering the cost of vessels deployed to services, overall fuel consumption, port call costs and cargo handling costs. Literature on the LSNDP is

[1] Conference paper published in proceedings of *LOGMS 2012* (2012)

quite scarce (Brouer et al., 2012) compared to related maritime shipping transportation problems, but a surge in publications over recent years show increased interest in the LSNDP. The works of Agarwal and Ergun (2008); Alvarez (2009); Reinhardt and Pisinger (2011); Plum (2010); Brouer et al. (2012) reveal that the liner shipping network design problem is a very complex optimization problem, where current mathematical formulations and state-of-the-art exact solution methods cannot scale to realistic sized problem instances at the time of writing. One heuristic approach has been applied to large scale instances in (Alvarez, 2009; Brouer et al., 2012). A core concept in liner shipping is the transshipment of containers. More than 50% of cargoes are transported on more than one service from origin to destination. This means that we can model the LSNDP with an underlying multicommodity flow problem (MCF). Alvarez (2009) identifies the excessive time used for solving the MCF to evaluate a given network configuration as a bottleneck in local search methods. As a result, within reasonable computation time the tabu search by Brouer et al. (2012) only performs a limited search of the solution space of large scale instances.

In this paper, we present a matheuristic for solving the LSNDP. Matheuristics are an emerging field within optimization and are defined as methods exploiting the synergies of mathematical programming and metaheuristics (Maniezzo et al., 2009). The domain is wide and includes the use of mathematical programming techniques in a heuristic variant as well as deploying mathematical programming methods within a metaheuristic framework (Maniezzo et al., 2009). In the present paper we use mathematical programming to explore our neighborhood defined as the solution space of a mixed integer program designed to capture the complex interaction of the cargo allocation between routes.

One of the first approaches of using this technique was Franceschi et al. (2006) for the Distance-Constrained Capacitated Vehicle Routing Problem. The method has also been explored for the Split Delivery Vehicle Routing Problem by Chen et al. (2007), the Split Delivery Vehicle Routing Problem with minimum delivery amounts by Gulczynski et al. (2010), by Archetti et al. (2010) for the The Split Delivery Capacitated Team Orienteering Problem and lately in Gulczynski et al. (2011) for the Periodic Vehicle Routing Problem. In all cases the matheuristic solution method combining local search with an integer program as neighborhood has proven very successful compared to other state-of-the-art heuristics.

In the present paper we exploit this technique in a metaheuristic framework for the LSNDP. We make four main contributions: We present a construction heuristic for the LSNDP by transforming the problem into an instance of the multiple quadratic knapsack problem. Secondly, an improvement heuristic is applied to the solution of the construction heuristic. The improvement heuristic is a large neighborhood search defined as a mixed integer program inserting and removing port calls from a single service. Thirdly, the heuristic makes use of estimation functions for the change in a large MCF, in order to avoid the bottleneck of solving a large scale MCF. Once moves are applied to a service the neighborhood of subsequent services is based on an optimal solution of the MCF in order to decrease the error of the evaluation functions. The MCF is resolved using an advanced warm start basis and column generation, decreasing solution times significantly. Lastly, we present computational results using the matheuristic to solve instances of the benchmark suite *LINER-LIB 2012*. The results are the first results for the benchmark suite enforcing a strict weekly frequency for all services. The implementation is quite fast in solving the instances and is able to improve the constructed initial solution with 60-400% yielding profitable networks for tvelwe out of eighteen solved instances. Additionally a single test case has been constructed to evaluate the matheuristic as a decision support tool, where the initial solution is an existing network and we allow optimization of a subset of the services considering the flow in the entire network. Results are promising for this approach.

The outline of the paper is as follows. In Section 5.2 we review the literature on liner shipping network design. Section 5.3 describes the individual components of our matheuristic. Section 5.4 presents the design of the complete algorithm for the matheuristic. Section 4.5 presents computational results for the matheuristic for LSNDP using the benchmark suite *LINER-LIB 2012* followed by a brief conclusion and perspectives for future work.

5.2 Literature on the LSNDP

Brouer et al. (2012) give an introduction to the LSNDP focusing on mathematical modelling of the business domain and the introduction of a benchmark suite of LSNDP problems.

Christiansen et al. (2004) review the field of operations research within shipping in general and a good introduction to the LSNDP may be found in Christiansen et al. (2007). Recently, Kjeldsen (2011) published a classification scheme for routing and scheduling problems within liner shipping reviewing and classifying 24 references. The LSNDP was initially studied by Rana and Vickson (1991) as a Mixed Integer Program (MIP) for a multiple container-ship problem without transshipment and where vessels return to the origin node empty. Benders decomposition principle divides the MIP into an integer network subproblem (INS) and a cargo allocation problem (CAS). Results are reported for 10-20 ports and three vessels.

In recent literature several variants of the LSNDP have been presented. Fagerholt (2004) developed a model and solution method for a regional carrier along the Norwegian coast. The model assumes the carrier loads at a single port and finds optimal routes of vessels to service the unloading facilities. The problem may be dealt with as a VRP problem, given that a designated depot is known and transhipments are not allowed. The solution method is based on complete enumeration solved by a MIP solver. Similarly, Karlaftis et al. (2009) solved a problem for the region of the Aegean sea using a genetic algorithm. These models do not deal with the important concept of transhipments at multiple ports and the resulting interaction between different services.

The simultaneous ship scheduling and cargo routing problem (SSSCR) by Agarwal and Ergun (2008) is based on a time-space network with each port represented on 7 consecutive weekdays. This construction allows non-simple cycles with multiple visits to a port on different weekdays. Computational results were reported for three different heuristics exploiting the separability of solving the route generation problem and the MCF: a greedy cycle-generation heuristic, a column generation based heuristic and a two-phase Benders decomposition algorithm. Results are reported for 6, 10, 15 and 20 ports with up to 100 ships and 114 demands. The Benders decomposition algorithm is the best performing heuristic of the three, but no optimality gap can be reported for the solutions found as a standard branch-and-bound procedure is invoked for both the Benders, and the column generation algorithm without a complete set of routings. An important limitation of the SSSCR is that it allows transshipments at no cost.

Reinhardt and Pisinger (2011) presented the LSNDP for a multiple container ship problem with separate routings for each vessel accounting for transshipment costs between routes. The model allows pseudo-simple cycles, where multiple visits are allowed to one port on a service. A branch-and-cut algorithm is applied to the problem and computational results are reported for 15 ports and up to 6 vessels. Instances of up to 10 ports using 3 vessels are solved to optimality, but the solutions for the larger test instances have a gap of up to 25%.

Alvarez (2009) presented the joint routing and fleet deployment model for the LSNDP. The model accounts for transshipment costs and the option of laying up or forward leasing vessels not in use. The model is separable into a service generating problem and a MCF. A service consists of a port call sequence, a number of vessels deployed and an average sailing speed. The overall objective is to maximise the revenue of cargo transported, while considering operational cost of the fleet-, fuel-, transshipment-, and port call-cost. The model is the first to incorporate routings with different speeds in order to optimize on the fuel consumption in the network. Exact solutions are obtained for a six port instance using a MIP solver. Alvarez (2009) describes a tabu search heuristic to solve the problem which is applied to a 120 port instance with a full demand matrix.

Recently, Meng and Wang (2011) presented a mixed integer programming model with the objectives to select among a set of predefined candidate shipping routes, and to select ship deployment to the chosen routes while considering the cargo allocation of full and empty containers regarding the weekly frequency constraint. The model aims at providing decision support on existing routes as well as user-defined candidate routes by minimizing the fixed cost of deployment and the variable costs associated with full and empty containers. The model is solved using CPLEX and numerical results are presented for 60 candidate shipping lines, eight vessel types and 600 commodities.

Brouer et al. (2012) presents a reference model for the LSNDP similar to the model of Alvarez (2009) accounting correctly for transshipments on butterfly routes. A heuristic column generator generating routes using a MIP is used to solve the benchmark suite *LINER-LIB 2012*. The largest instance solved contains 111 ports and 4000 demands.

In light of the literature published on the LSNDP exact solution methods are presently not able to solve large scale instances to optimality. Heuristic methods published are often based on route generation in a branch-and-bound framework with the exception of Alvarez (2009); Brouer et al. (2012), where the overall framework is a local search with route generation as the underlying method for producing a new candidate solution. The present paper aims at improving an existing solution by modifying the current services with insertion and removals of port calls using a multiple quadratic knapsack problem to create an initial solution and a simple method for generating new services to diversify the search. To the best of our knowledge, it is the first heuristic presented for LSNDP using a mixed integer program to define the neighborhood of insertions and removals. The literature identifies a bottleneck in evaluating a candidate solution by solving a large scale arc flow formulation of the MCF using a MIP solver. To increase performance of evaluation we apply a novel method exploiting a warm started delayed column generation algorithm on a path flow formulation of the MCF to evaluate a candidate solution.

5.3 A matheuristic for the liner shipping network design problem

An instance of the liner shipping network design problem consists of:

- A set of ports P.
- A set of demands K, where each demand has an origin, $O_k \in P$, and a destination, $D_k \in P$.
- A set of vessel classes A and a quantity of each vessel class N_a. Each vessel v belongs to a given vessel class a specifying its capacity C_a, minimum and maximum speed limits, bunker consumption per nautical mile and a weekly sailing distance W_d^a. The weekly sailing distance is based on the design speed of the vessel, where fuel consumption is optimized.
- A distance table of the direct distance d_{pq} between all pairs of ports $p, q \in P$.

A solution to the liner shipping network design problem is a set of services S. A service is a cyclic route visiting a set of ports $P' \subseteq P$. The service may be non-simple. The rotation time is the time needed to complete the cyclic route including a day for each port call en route for cargo handling. Depending on the vessel class a a minimum (T_{min}^a) and maximum (T_{max}^a) rotation time in weeks may be defined. It is common in liner shipping to offer a regular service with a weekly frequency. The weekly frequency of port calls is obtained by deploying multiple vessels to a service sailing one week apart. Let n_s^a be the number of vessels of vessel class $a \in A$ deployed to service $s \in S$ to maintain a weekly frequency. A service carries a set of demands $k_s \subseteq K$ either by serving both O_k and D_k or by serving either O_k or D_k and a designated transshipment port G_k valid for transshipping demand $k \in K$.

5.3.1 Mathematical model

A reference model for the LSNDP is provided in Brouer et al. (2012). The variables of this model are rotations defining a service with a vessel class a, the number of vessels deployed and a speed. The rotations generated by the heuristic column generator in Brouer et al. (2012) have a frequency corresponding to the number of vessels, the speed and the distance travelled allowing for (bi)weekly frequencies for vessel classes below 1200 FFE and weekly frequencies for all remaining vessel classes. Additionally this (bi)weekly frequency has a span of 5-7 days frequency for weekly frequency and 10-14 days for bi-weekly frequency. In the work on the LSNDP the question of weekly frequency is disputed as one research trend is towards a planning horizon introducing some

flexibility on the frequency as in Alvarez (2009); Brouer et al. (2012); Reinhardt and Pisinger (2011) whereas Agarwal and Ergun (2008); Plum (2010) enforce strict weekly frequency. In the present paper we focus on a strict weekly frequency for all vessel classes, which gives a more restricted solution space than in the heuristic presented in Brouer et al. (2012). However, the only difference is in the solution space of the rotations and we adhere to the objective function and the constraints of the reference model. The objective accounts for the daily running cost of vessels deployed, the fuel consumption, canal costs and port call cost. A revenue is obtained for the cargo transported and a penalty is incurred for cargo not transported. Finally, a cargo handling costs for loading, unloading and transshipments is charged. Note that we are solving a minimization problem, which means that a negative objective value represents a profit.

5.3.2 Algorithmic overview

The matheuristic creates an initial solution using a greedy construction heuristic. The construction heuristic returns a set of services, S, that are iteratively improved using a MIP for each service to indicate a set of port insertions and removal of each individual service. The MIP returns a set of moves and the resulting candidate solution is evaluated using a warm started delayed column generation algorithm and a simulated annealing scheme decides whether the candidate is accepted. This phase of the heuristic fine tunes a given solution. The composition of the set of services is important in a high quality solution and we subsequently apply a local search on the composition of the set of services S. Note that the reference model is defined as a minimization problem and a negative objective value is a profitable network. An algorithmic overview is illustrated in figure 5.1.

5.3.3 Generating an initial solution using a greedy construction heuristic

We obtain an initial solution to the LSNDP by constructing a set of services in which we place a set of predefined port calls in order to transport the demand. The construction heuristic transforms an instance of the LSNDP into an instance of the multiple quadratic knapsack problem, where a service corresponds to a knapsack and the items are port calls. It is quadratic in the sense that profit is obtained by adding port pairs to the services in order to transport demand.

The service set problem is based on a subdivision of the available fleet into a set of services S constituted by subsets $S_a \subseteq S$ according to vessel classes. It is desirable to have services of varying duration within an interval $[T^a_{min}; T^a_{max}]$. A random integer $h \in [T^a_{min}; T^a_{max}]$ is selected and a service s with $n^a_s = h$ of h weeks duration (an h-week rotation requires h vessels) is added to the set of services S. Set $v = N_a$. After creation of a new service $s \in S_a$, v is updated by subtracting h from v. This process is repeated until $v \leq T^a_{max}$. If $v \geq T^a_{min}$ the final service is created with $h = v \in [T^a_{min}; T^a_{max}]$ otherwise we add h vessels to the previously created service s' possibly exceeding T^a_{max}. If $n^a_{s'} \geq 2 \cdot T^a_{min}$ we split the service into two services each with $n^a_s = h/2$. After this procedure, a set of services S is defined, a vessel class a and a number of vessels n^a_s is assigned to each service s. The subdivision of the fleet into services means that the initial solution attempts to assign the entire fleet to services.

Next we define a set of port calls to place in the services. Each port can be defined as a main port or an outport. The initial solution is limited to the creation of simple cycles. A port call may be placed only once in each service, but in m_p services. Outports have $m_p = 3$, whereas main ports have $m_p = 10$. There is no constraint requiring all port calls placed in the set of services.

The profit of transporting a demand from port $i \in P$ to $j \in P$ is the revenue r_k obtained by the transport subtracted the loading and unloading cost (c^i_l and c^i_u respectively) of the container en route. A demand transported with no transshipments will have net revenue $\rho_{O_k D_k} = (r_k - c^{O_k}_l - c^{D_k}_u)$ for one unit of k. As described in the introduction more than 50% of the demands are transshipped resulting in a MCF. The multiple quadratic knapsack problem does not consider this MCF, and in order to model transshipments the demand matrix is transformed such that each demand is represented by a direct demand and a demand transshipped at a designated

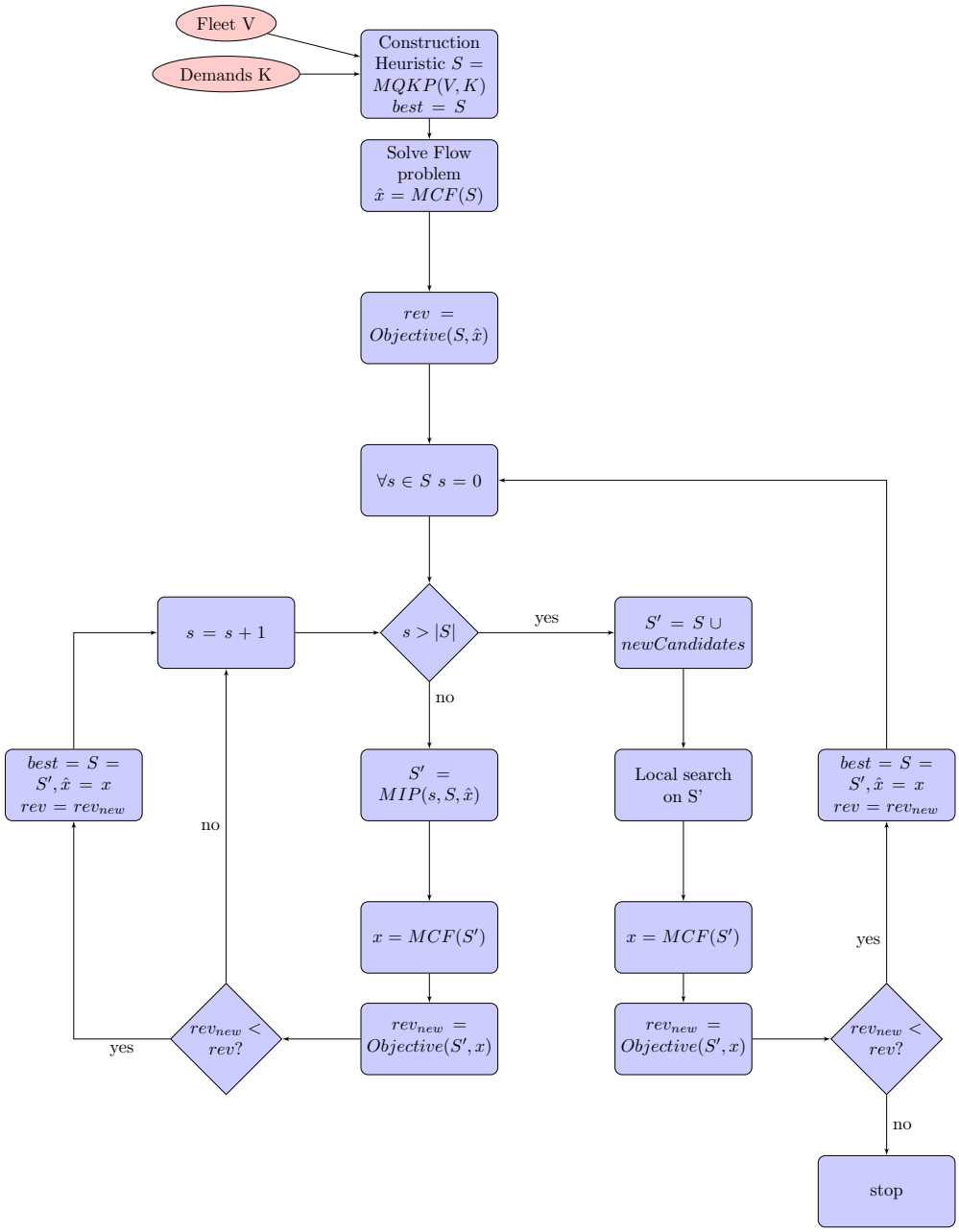


Figure 5.1: A flow chart of the matheuristic.

transhipment port G_k, where $\rho_{O_k G_k} = (\frac{1}{2} r_k - c_l^{O_k} - c_u^{G_k})$ and $\rho_{G_k D_k} = (\frac{1}{2} r_k - c_l^{G_k} - c_u^{D_k})$. This is a simplifying assumption fixing a single transshipment port for each demand to incorporate interaction between services in the construction heuristic. The subsequent improvement heuristic will have no restrictions on transshipment facilities. A port call cost, c_p^a, is associated with a port call depending on the vessel class and a sailing cost is associated with each port pair, c_{pq}^a.

The construction heuristic is a greedy parallel insertion heuristic. The services are seeded with a random number $l \in \{1; 3\}$ of ports $p \in P$. The seeding is either by random or by selecting a port $p \in P$ and a transhipment port $q \in P$ matched to p. The construction heuristic is based on parallel insertion by a *football teaming principle* i.e. the services take turn at choosing the next port to call. We apply parallel insertion in order to disperse the attractive port call

combinations throughout the network. A greedy choice of the most revenue generating port call is made between all feasible port calls with regards to route duration. Feasibility of a given port call is estimated using best insertion in order to respect the weekly frequency constraint, requiring the distance of a route $D_s \leq W_d^{a(s)}(n_s^{a(s)} - (\frac{|P_s|}{7}))$, where $|P_s|$ are the number of port calls in service s, $a(s)$ is the vessel class $a \in A$ deployed to service $s \in S$. The number $\frac{|P_s|}{7}$ accounts for the accumulated port stay of the service, where 24 hours are used in each port of call effectively decreasing the time left for actually sailing a round trip. The actual routing with regards to distance and capacity utilization is improved using a local search based on simulated annealing and two-opt after assignment of port calls to services by the greedy construction heuristic. The initial solution may have unplaced port calls and excess vessels for services s, where the distance of a route allows this ($D_s \leq W_d^{a(s)}(n_s^{a(s)} - (\frac{|P_s|}{7}) - 1)$). Port calls as well as vessels may be included in the solution of the subsequent improvement heuristic. Finally, we apply standard column generation to the MCF of transporting the cargo on the resulting liner shipping network of the initial solution. The solution of the MCF is used to calculate the estimation function values of the improvement heuristic.

5.3.4 Improvement heuristic

Given a solution to the LSNDP x' with services S' serving demands $K' \subseteq K$ we introduce an integer program to estimate the effect of removing and adding port calls. We define:

- P^s is the set of nodes in the service $s \in S'$.
- $N^s \subset P \setminus P^s$ is the set of neighbors of a service $s \in S'$ defined as nodes within a certain geographical distance of nodes in P^s.

We have the variables:

- $\lambda_i = 1$ if item $i \in P^s$ is removed from service $s \in S'$, 0 otherwise.
- $\gamma_i = 1$ if item $i \in N^s$ is inserted in service $s \in S'$, 0 otherwise.
- $\omega_s \in \mathcal{Z}^+$ an integer variable indicating the number of vessels service s is expanded with. ω_s can be negative if less vessels are needed after removal of a port call.

We want to make an integer program that removes and inserts port calls in S', while considering an estimation of the duration of each service (the fleet deployment) and an estimation of the alternative flow of demands arising, when we remove/insert several port calls from/to S'. Routing the cargo is a MCF, but we cannot afford to evaluate the MCF in its entirety and hence we make some simplifying assumptions about rerouting the flow.

Estimation functions for the distance travelled by each service $s \in S'$

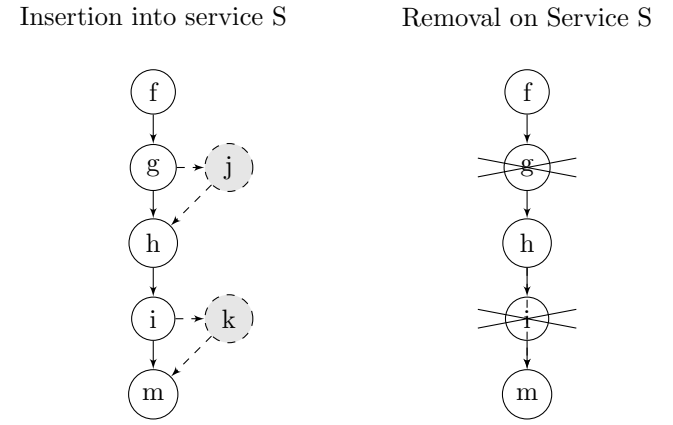
When inserting a port call the estimated distance increase is calculated by use of a best insertion heuristic. For each service $s \in S'$ we calculate the distance increase Δ_i^s for each $i \in N^s$. Likewise we calculate the decrease of distance Γ_i^s for every $i \in P^s$. For modelling the distance in-/decrease of insertions/removals we define the following constants and sets:

Δ_i^s : estimated distance increase for inserting item i in service $s \in S$ according to a best insertion method.

Γ_i^s : estimated distance decrease for removing item i from service $s \in S$ joining its predecessor with its successor.

E_s : set of edges used by the Hamiltonian cycle in service $s \in S$.

$D(s)$: current distance of the Hamiltonian cycle in service $s \in S$.



(a) Insertion of port call vertex. The dashed nodes j and k are considered for insertion

(b) Removal of a port call vertex. The crossed out nodes g and l are considered for removal

Figure 5.2: The neighborhood consists of a combination of insertion and removal moves

M_a : number of undeployed vessels of class a in the current service set S'.

n_s^a : number of deployed vessels of class a to service $s \in S'$.

C_v^a : cost of deploying a vessel of type $a \in A$.

Estimation functions for the change in the multicommodity flow

Whenever a MIP is solved for some $s \in S'$ we estimate the effect on the flow in the network by solving shortest path problems on the residual capacity of the network. The quality of the flow solution depends on:

1. The number of transshipments performed overall in the network.
2. The capacity installed compared to the demand for flow.

We define the following estimation functions:

$\Theta(i)$: estimated value of inserting a node $i \in N^s$ in the best insertion position identified when calculating the distance.

$\Upsilon(i)$: estimated value of removing a node $i \in P^s$.

$\Psi(i)$: estimated value of reinserting a node $i \in P^s$ by best insertion limited to insertions two port calls away from the current position of i in s.

Graph topology

In order to estimate the change of the network flow a graph $G = (V, E)$ defined by the residual capacity is constructed, representing the solution x' with services S' and commodity allocation K' mapped onto the network by solving the MCF on S'. Let:

- $|s|$ denote the number of unique ports in s and let $|P^s| = m$ denote the number of port calls in a rotation r^s for s, $|P^s| = m \geq |s|$.
- r^s be a rotation defined by the port sequence $p_1^s, p_2^s, \ldots, p_m^s$.
- V_p be the set of port vertices.

- V_{r^s} be a set of vertices representing the port call sequence $p_1^s, p_2^s, \ldots, p_m^s$ for rotation r^s.
- $V_r = \bigcup_s V_{r^s}$ be the set of rotation vertices representing all port calls by all rotations.
- $V = V_p \cup V_r$ be the set of vertices.
- $E = E_l \cup E_d \cup E_v \cup E_t$ be the set of edges, where
  1. $E_l = \{(p, v) | p \in V_p, v \in V_{r^s}\}$ is the set of load edges representing a departure from port p to the rotation r^s.
  2. $E_d = \{(v, p) | v \in V_{r^s}, p \in V_p\}$ is the set of discharge edges representing an arrival at port p from the rotation r^s.
  3. $E_t = \{(v, u) | v \in V_{r^s}, u \in V_{r^{s'}}\}$ is the set of transshipment edges representing a transshipment between rotations r^s and $r^{s'}$ defined for every (v, u) where $p(v) = p(u)$, where $p(w)$ denotes the physical port of the port call w.
  4. $E_v = \{(v, u) | v, u \in V_{r^s}, v = p_h^s, u = p_{((h+1) \bmod m)}^s\}$ is the set of voyage edges representing a voyage between two consecutive port calls in r^s defined $\forall \quad h \in \{1, 2, \ldots, m\}$.
- C_e be the capacity of edge $e \in E$, where $C_e = \infty$ for $e \in E_l \cup E_d \cup E_t$ and $C_e, e \in E_v$ be the residual capacity of edge e after flow assignment of the MCF onto S'.
- c_e be the edge cost, where $c_e = 0, e \in E_v$ (as the cost is on the vessel) and let $c_e = c_l^p, e \in E_l$ and $c_e = c_u^p, e \in E_d$ be the cargo handling cost of loading or unloading a container at port $p \in V_P$, where p is either the source or the target of the edge respectively. For the transshipment edges $c_e = c_t^p$, where p is the physical port of the port calls of the source and the target of edge e.

An example of a hub and spoke network with one hub, C, and five spokes (A, B, D, E, F) and 3 rotations is illustrated in Figure 5.3. The rotations are $A \to B \to C \to A$ (index 1), $C \to A \to C \to D \to C$ (index 2), $F \to E \to C \to F$ (index 3). Each rotation vertex has a load and a discharge edge to the physical port and a voyage edge to the next port en route. A demand from A to F transships at C using rotations 2 and 3: $A \to A2 \to C2 \to C3 \to F3 \to F$ (edge colours green and blue in Figure 5.3) or using rotations with indices 1 and 3: $A \to A1 \to B1 \to C1 \to C3 \to F3 \to F$ (edge colours red and blue in Figure 5.3). The load/unload cost and the transshipment cost is accounted for by the path cost. The load/unload is distinct from the transhipment as some ports have a different(lower) cost for transshipment than for a load combined with an unload. Note, that multiple visits to a port on a service results in multiple vertices. To adhere to the reference model a single port is allowed to be called twice.

The estimated value of insertion - $\Theta(i)$

When we insert a port call vertex $i^s \in N^s$ with corresponding port vertex $i \in V_p$ in the position between nodes h^s and l^s the demands of the set $K_i = \{k \in K | i = O_k \vee i = D_k\}$ become eligible for transport using service $s \in S'$. Solving a shortest path problem on G' where $V'_{r^s} \cup \{i^s\}$, and $E'_v = E_v \setminus \{(h^s l^s)\} \cup \{(h^s i^s), (i^s l^s)\}, E'_l = E_l \cup \{(ii^s)\}, E'_d = E_d \cup \{(i^s i)\}$ will identify for each $k \in K_i$ whether there is an (improved) path for k in G' in terms of transshipment costs (TC), the increase of revenue in demand transported (RK) and the capacity available. The estimated value $\Theta(i)$ should account for in-/decrease in transshipment cost, in-/decreased revenue of the flow, and increase in port call cost: $\Theta(i) = TC(G', K_i) - TC(G, K_i) + RK(G') - RK(G) - c_i^s$.

The estimated value of removal - $\Upsilon(i)$

When a port call vertex $i^s \in P^s$ is removed between nodes h^s and l^s, commodities of the set K_i transported on s must be rerouted or omitted. Define $K_i^s = \{k \in K_i | k \text{ is transported on } s\}$. $\Upsilon(i)$ estimates rerouting K_i^s in the remaining network by solving a shortest path problem on

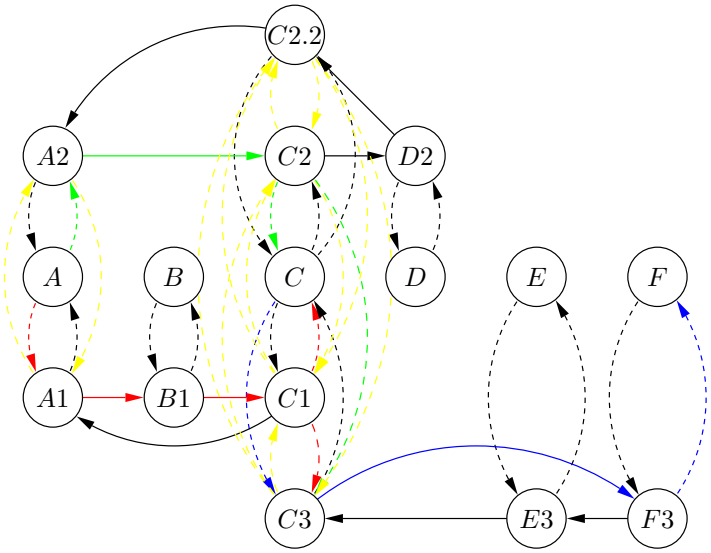


Figure 5.3: Example network with 6 ports (A, B, C, D, E, F) and 3 rotations. C is the linking hub. Continuous arcs are voyage edges E_v, whereas dotted arcs are load, discharge and transshipment edges $E_l \bigcup E_d \bigcup E_t$. E_t are yellow to set them apart from load/unload edges. The red (A, A1, B1, C1, C)-and green (A, A2, C2, C) paths are two distinct ways to travel from A to C. The blue (C, C3, F3, F) path is the only path from C to F.

$G' = (V_{r^s} \setminus \{i^s\}, E_v \setminus \{(h^s i^s), (i^s l^s)\} \cup \{(h^s l^s)\})$. G' will identify for each $k \in K_i^s$ whether there is an alternative path in the network. The estimated value $\Upsilon(i)$ should account for the in-/decrease in transshipment cost TC for each commodity $k \in K_i^s$ rerouted in G', and the decrease of revenue flow RK for omitted cargo and the decrease in port call cost.
$\Upsilon(i) = TC(G', K_i) - TC(G, K_i) + RK(G') - RK(G) - c_i^s$.

Lock sets

The estimation functions are used to make a MIP for a single service with the remaining services fixed. A solution to the MIP may result in several insertions and removals referred to as a *move* in the following. The estimation functions are based on performing a particular move without consideration of additional removals/insertions. In order to reduce the error of the estimation functions we define *lock sets* of a move constraining insertions/removals on port calls related to a move.

When inserting a port call i, a set of new commodities K_i may be transported. The origins and destinations of $k \in K_i$ should not be removed. The estimation function relies on the residual capacity of the remaining network. Insertions before bottlenecks introduced by the routing of K_i should be avoided. We define the set of *Insertion locks* on inserting $i \in N^s$ as $L(i^+)$. $L(i^+)$ place a lock on removal of origin/destination nodes ($i \in P^s$) for $k \in K_i$, and lock on insertion of nodes ($i \in N^s$) with best insertion position before bottlenecks introduced by routing K_i.

The total number of removals from a service is constrained to F_s. F_s is input by the user or dependent on the number of port calls on a service. F_s should be small to reduce the error of the distance decrease function Γ_i^s.

5.3.5 MIP formulation

The following MIP optimizes a single service and suggests a set of removals and insertions of port calls. The function $a(s)$ returns the vessel class assigned to service s.

$$max \sum_{i \in N^s} \Theta(i)\gamma_i + \sum_{i \in P^s} \Upsilon(i)\lambda_i - C_v^{a(s)}\omega_s \tag{5.1}$$

subject to: $D(s) + \sum_{i \in N^s} \Delta_i^s \gamma_i - \sum_{i \in P^s} \Gamma_i^s \lambda_i \leq$

$$W_d^{a(s)}(n_s^a - \frac{|P_s| + \sum_{i \in N^s} \gamma_i - \sum_{i \in P^s} \lambda_i}{7} + \omega_s) \tag{5.2}$$

$$\omega_s \leq M_{a(s)} \tag{5.3}$$

$$\sum_{i \in P^s} \lambda_i \leq F_s \tag{5.4}$$

$$\sum_{j \in L(i^+)} \gamma_j + \lambda_j - 2|L(i^+)|(1 - \gamma_i) \leq 0 \qquad \forall i \in N^s \tag{5.5}$$

$$\lambda_i \in \{0, 1\} \qquad \forall i \in P^s \tag{5.6}$$

$$\gamma_i \in \{0, 1\} \qquad \forall i \in N^s \tag{5.7}$$

$$\omega_s \in \mathcal{Z} \tag{5.8}$$

Sets	
P^s	The set of nodes in the service s.
N^s	The set of neighbors of service s.
$L(i^+)$	Lock set of inserting i in s.
Constants	
$C_v^{a(s)}$	The weekly cost of a vessel from vessel class a assigned to service s.
$M_{a(s)}$	The number of undeployed vessels of vessel class a assigned to service s in the current solution.
$W_d^{a(s)}$	The weekly sailing distance of a vessel of class a at design speed.
F_s	number of removals allowed on each service
Variables	
λ_i	1 iff $i \in P^s$ is removed from service s, 0 otherwise.
γ_i	1 if item $i \in N^s$ is inserted in service s, 0 otherwise.
$\omega_s \in \mathcal{Z}^+$	The number of vessels service s is expanded with.
Functions	
Δ_i^s	Estimated distance increase if i is inserted in s.
Γ_i^s	Estimated Distance decrease for removing item i from s.
$\Theta(i)$	The estimated value of inserting i in s.
$\Upsilon(j)$	The estimated value of removing j from s

Table 5.2: Overview of Sets, variables, and functions used in the MIP

The objective function (5.1) maximises the benefit obtained from removing and inserting several port calls accounting for the estimated change of revenue, transshipment cost, port call cost and fleet cost. The number of vessels needed to maintain weekly frequency on the service after insertion/removal is estimated in Constraints (5.2) accounting for the duration of the port stay given insertions/removals. Constraints (5.3) ensure that the solution does not exceed the available fleet of vessels. The set of nodes that are affected by the insertion move are fixed by Constraints (5.5). Constraints (5.4) ensure that we can only remove F_s nodes from the service.

To diversify the search using the MIP the geographical distance to the ports included in the neighborhood N_s is increased for every iteration, where no improvement is found. As the ports on a service are often geographically close to each other some ports in the service have neighboring ports in common. For this reason the neighborhood size varies a lot for the service even when increasing the geographical distance. The increase in geographical distance is inspired by Variable

Neighborhood Search (VNS)(Mladenović and Hansen, 1997), where the neighborhood depth is increased iteratively.

We introduce a number of taboos in each service to avoid cycling between solutions. This is to ensure that ports inserted in the previous iteration cannot be removed in the subsequent MIP for a specific service. Likewise, ports removed in the last iteration are not considered for reinsertion in the following iteration.

5.3.6 Reinsertion neighborhood

It is important to be able to construct non-simple routes (butterfly rotations), that revisit a single port twice. Butterfly routes are very important to utilize the capacity efficiently in a liner shipping network. A reinsertion neighborhood is defined to construct butterfly routes by evaluating reinsertion of port calls on every service in S. The reinsertion neighborhood loops through the services in S and identifies the most promising port call to reinsert based upon a rating. The method identifies every port call, where the outgoing voyage arc has no residual capacity. Each of the candidate port calls on a service are rated based on the residual demand. A score of 1 is obtained for each residual demand originating at the candidate port and an additional score is obtained if the destination of the commodity is found on the service. The port obtains an additional score of 3 if the port is a designated hub port. Based on the rating the port with the highest rating is reinserted using the best insertion heuristic requiring the reinsertion to be a minimum of one port call away from the original port call. The reinsertion is immediately evaluated and accepted if it improves the objective value or increases the amount of cargo transported. An overview of the reinsertion neighborhood is found in Algorithm 1

Algorithm 1 $EvaluateReinsertion(S', x', \tilde{K})$

Require: A solution S of the LSNDP $(S, \tilde{K}, x)$

$\hat{S} \leftarrow S'$
$\hat{x} \leftarrow x'$
for all $s \in S'$ **do**
  $Candidate \leftarrow \emptyset$
  for all $p_i^s \in s$ **do**
    if $flow(p_i^s, p_{i+1}^s) - Capacity(p_i^s, p_{i+1}^s) = 0$ **then**
      $Candidate \leftarrow p_i^s$
      $rating_{p_i^s} \leftarrow AssignRating(\tilde{K}, p_i^s, S)$
    end if
  end for
  if $Candidate$!= $\emptyset$ **then**
    $R \leftarrow MaxRating(Candidate)$
    $\bar{S} \leftarrow ApplyReinsertion(R, s)$
    $\bar{x} \leftarrow \Delta MCF(\bar{S}, K)$
  end if
  if $obj(\bar{x}) \leq obj(x') \vee trans(\bar{x}) > trans(x')$ **then**
    $x' \leftarrow \bar{x}$
    $S' \leftarrow \bar{S}$
    $\tilde{K} = CalcResidualDemand(S', x')$
  end if
end for
return (S', x')

5.3.7 Local search

The solution space of an instance of the LSNDP has several sub-neighborhoods that must be considered in a search for a high quality solution. The main sub-neighborhoods are listed below:

1. The individual port calls on each service.
2. The selection of the butterfly port.
3. Transhipment points.
4. Deployment of a number of vessels to a service (i.e. the duration assigned)
5. Average speed of the vessels on a service.
6. The composition of the services.
7. The number of services in the solution.
8. Matching vessel classes to services.

The MIP and the reinsertion neighborhood target sub-neighborhood 1, 2, 3, and 4. In the application of a move sub-neighborhood 5 is considered as well. An adjustment of the number of vessels is performed relative to the feasibility and cost of assigning one less or one more vessel to the service. If it is cheaper and feasible to add/subtract a vessel from a service, this optimization is performed, when applying the move. This procedure also adjusts the speed of a service, although the design speed will be the cheapest choice if it can be matched with the duration and the number of vessels assigned to a service.

A weakness of the MIP neighborhood is that it is not able to adjust the number of services. The number of services in the initial solution of the construction heuristic is not likely to be optimal. At the same time the composition of services is not considered for example if two services are roughly identical and therefore both underutilized with regards to their capacity.

Therefore, a local search is performed on the set of services after every fourth loop of the MIP neighborhood application to the set of services.

The local search rates every service according to its average utilization of the capacity on all voyage edges. In increasing order of utilization removing a service is evaluated. Up to two services are removed using this procedure. The limitation is imposed to decrease the size of the move. The residual demand is updated for the solution with services removed and a procedure to add new services based on the residual demand is invoked. A single service is added for every vessel class with undeployed vessels with the largest matching residual demand respecting feasibility of the capacity and duration of serving the demand with the vessels available.

Lastly, the MIP sub-neighborhood cannot assign a different vessel class to a service for example a service which is fully utilized and in general needs to be upgraded to a larger vessel class with more capacity and likewise a service with a medium average utilization may become more profitable by assigning a smaller vessel class to the service. In the current implementation we do not consider swapping vessel classes between services although it would be very interesting to investigate the performance of such a neighborhood. The neighborhood would require extensive analysis to find good candidate services for swapping vessel classes and would involve reflowing the cargo upon each swap to evaluate the move. It is hence not trivial to implement.

5.4 Algorithm

Algorithm 2 gives an overview of the matheuristic. The initial solution is constructed by the greedy parallel insertion $GreedyLSNDP(I)$ of an instance I in line 1. The initial solution does not contain exact evaluation of the flow and hence we explicitly solve the resulting MCF in line 2. The problem $MCF(S', K)$ is solved using standard column generation to construct a graph of

Algorithm 2 MatHeuristic(I)

Require: An Instance I of the LSNDP (S, P, A, D, K)

1: $S' \leftarrow GreedyLSNDP(I)$
2: $x' \leftarrow MCF(S', K)$
3: $iter \leftarrow 0$
4: $temp \leftarrow temp_0$
5: $IMPROVE \leftarrow true$
6: **while** $(IMPROVE \wedge (temp > 0))$ **do**
7: $IMPROVE \leftarrow false$
8: **for all** $s \in S'$ **do**
9: **if** $obj(x') \leq obj(\hat{x})$ **then**
10: $SaveBest(S', x')$
11: **end if**
12: **for all** $i \in N^s$ **do**
13: $(L(i^+), \Theta(i)) \leftarrow CalcInsert(i, S')$
14: **end for**
15: **for all** $i \in P^s$ **do**
16: $\Upsilon(i) \leftarrow CalcRemoval(i, S')$
17: **end for**
18: $MIP \leftarrow constructMIP(s, P^s, N^s, \Theta, \Upsilon, \Psi, \bigcup_{i \in N^s \cup P^s} L(i^+), \bigcup_{i \in P^s} L(i^2))$
19: $M \leftarrow solve(MIP)$
20: **if** M != $\emptyset$ **then**
21: $\bar{S} \leftarrow ApplyMoves(M, L(i^s_-), s)$
22: $\bar{x} \leftarrow \Delta MCF(S', K)$
23: **end if**
24: **if** $(obj(\bar{x}) \leq obj(x')) \vee (exp(\frac{obj(x') - obj(\bar{x})}{temp}) > random\,[0,\ 1\,[\,)$ **then**
25: $x' \leftarrow \bar{x}$
26: $S' \leftarrow \bar{S}$
27: $IMPROVE \leftarrow true$
28: **end if**
29: $temp \leftarrow temp \cdot 0.95$
30: **end for**
31: **if** $iter \mod 2 = 0$ **then**
32: $(S', x') \leftarrow EvaluateReinsertion(S', x')$
33: **end if**
34: **if** $iter \mod 5 = 0$ **then**
35: $S' \leftarrow LocalSearch(S', x')$
36: $temp \leftarrow temp * 10$
37: **end if**
38: **end while**
39: **return** $(\hat{S}, \hat{x})$

the residual capacity and mapping the current flow to specific services. In every iteration of the for loop the solution is evaluated according to the best found solution (line 9).

The improvement heuristic loops over the set of services S. The estimation functions and lock sets for s are calculated in lines 12-17. The MIP (5.1)-(5.8) for s is constructed and solved in lines 18-19.

The solution is evaluated by resolving the new MCF in line 22. $\Delta MCF(S', K)$ is a column generation algorithm for the MCF using a warm start basis. The basis consists of all commodities and services not directly affected by the moves identified by the MIP. The algorithm $\Delta MCF(S', K)$ is expected to decrease solution times, but the performance will depend on the number of commodi-

ties affected by the move and also the number of moves applied. If the solution is improved the new solution is saved in line 24-26 before the next MIP is calculated for the following $s \in S'$. If the solution is not improved a simulated annealing scheme is invoked, to accept worse solutions with a decreasing probability as the search progresses in lines 24-27. The simulated annealing scheme should help the algorithm escape from local minima. In every second loop over the set of services S a reinsertion neighborhood is invoked in order to evaluate the construction of butterfly services in line 32. In every fourth loop a local search is performed on the set of services evaluating the removal of underutilized services and adding new services based on the residual demand and the excess fleet (line 35). The algorithm terminates, when there is no improvement through a loop or the temperature of the simulated annealing is zero. The best found solution is returned in line 39.

5.5 Computational Results

The matheuristic was tested on the benchmark suite *LINER-LIB 2012* presented in Brouer et al. (2012). Table 5.3 gives an overview of the instances, which may be found at *http://www.or.man.dtu.dk/English/researc*

Category	Instance and description	Ports	Dmnds
Single hub instances	**Baltic** Baltic sea with Bremerhaven as hub	12	22
	WAF West Africa with Algeciras as hub to West African ports	19	38
Multi hub instances	**Mediterranean** Mediterranean with Algeciras, Tangier, Gioia Tauro as hub	39	369
Trade lane instances	**Pacific** (Asia-US West)	45	722
	AsiaEurope Europe, Middle East and Far east regions	111	4000
World instances	**Small** 47 Main ports worldwide identified by Maersk Line	47	1764
	Large 197 ports - the majority of ports serviced directly by Maersk Line	197	9630

Table 5.3: The instances of the benchmark suite with indication of the number of ports and the number of distinct origin-destination pairs. The instances may be found at *http://www.or.man.dtu.dk/English/research/instances*

The matheuristic is coded in `C++` and run on a linux system with a *Intel(R) Xeon(R) X5550* CPU at *2.67GHz* and *24 GB* RAM.

The following sections analyze the performance and the results of the algorithm. The first section analyzes the execution time required to solve the MIP neighborhood with growing instance size and the speed up of using delta column generation, when evaluating the neighborhood. The next section describes the results seen in relation to the results presented in Brouer et al. (2012) because our solution space is more restricted due to a required weekly frequency in the matheuristic for LSNDP. This makes it difficult to compare results. The final section presents the computational results for the benchmark suite *LINER-LIB 2012* with a weekly frequency and discusses the results obtained.

5.6 Performance of delta column generator and MIP neighborhood

In this section the performance of solving the MIPs to decide on the moves made to each service is evaluated. Secondly, the performance of evaluating a given solution for the flow transported using a warm started delayed column generation algorithm to solve the underlying MCF problem is presented.

In Table 5.4 the performance of the MIP neighborhoods are reported. The focus is on the execution time of building the MIP in terms of calculating the estimation functions and solving

the resulting MIP. The table shows that calculating the estimation functions and building the MIP is very fast. At the same time the MIPs are very small and solved fast both for small, medium and large cases. The growth in the neighborhoods is hence a linear growth with the number of services that increase with the instance size. The aim of the heuristic was exactly to achieve a linear growth in evaluating the neighborhood of services, such that it is fast to decide on the best moves for each service with regards to inserting and removing a number of port calls for each service considering the network flow as a whole.

Instance	av. time build (sec.)	av. time MIP (sec.)	av. vars	av. rows
Baltic	0.00012	0.00748	9.26	10.53
WAF	0.00036	0.00946	11.19	12.27
Mediterranean	0.00100	0.00972	19.32	19.81
Pacific	0.00390	0.00902	19.11	20.44
WorldSmall	0.01260	0.00600	17.41	18.72
AsiaEurope	0.02100	0.00820	19.93	21.33

Table 5.4: Av. time build: Time to build MIP (estimation functions), **av. time MIP**: Time to solve MIP , **av. vars**: Average number of variables (binary),**av. rows**: average number of rows.

In Table 5.5 the performance of an average run of the delta column generator is reported. The delta column generator is designed to exploit that inserting/removing a few port calls only invalidate a small fraction of the current optimal basis for the network. Therefore, we are able to reuse the optimal basis to a large extent. The table shows the time to solve the initial full multicommodity flow problem and the subsequent columns show the average solution time for the delta column generator as the maximal, minimum and average time throughout a single run of the algorithm for each instance. The result show a speed up of a factor 2-10 on average for evaluating a given solution with regards to the transported commodities and their transhipment costs.

Instance	full sec.	max sec.	min sec.	av. sec	av. speedup	max speedup
Baltic	0.0040	0.004	0.000	0.0015	2.67	–
WAF	0.0240	0.008	0.004	0.0045	5.33	6
Mediterranean	0.3760	0.050	0.016	0.0314	11.97	23
Pacific	3.6122	1.528	0.072	0.7178	5.03	50
WorldSmall	24.5400	12.316	0.024	2.7800	8.80	1022
AsiaEurope	147.9612	60.667	0.836	19.7500	7.49	177

Table 5.5: full sec.: Time to solve full lp, **max sec.**: maximum time for a warm start , **min sec.**: min sec for a warm start, **av. sec**: average time for warm start. **av. speedup** is the solution time for solving the full MCF divided by the av. seconds to solve the warm start. **max speedup** is the solution time for the full MCF divided by the minimum seconds to solve the warm started solution.

5.7 Comparing results

The problem solved with the matheuristic for LSNDP uses the reference model presented in Brouer et al. (2012) and adheres to the same set of general constraints and the same objective function. However, the routes generated by the matheuristic are restricted to have strict weekly frequencies for all vessel classes, whereas the heuristic column generator creates routes with both weekly and even biweekly frequencies for vessel classes under 1200 FFE. The rotations are allowed to deviate from a strict weekly frequency and may call as often as every five days for weekly frequencies. Vessel classes below 1200 FFE are allowed to have biweekly frequencies deviating to call as often as every ten days. This means that the network generated with the heuristic column generator has a less restricted solution space and may deploy the capacity in a different manner than the matheuristic for the LSNDP is able to. A direct comparison with the solutions and objective values

for the heuristic column generator is not meaningful as the best solutions found are not feasible solutions for the matheuristic. As a result the overall revenue obtained for the matheuristic and the reference model with strict weekly frequencies are expected to be lower. Please note, that we are solving a minimization problem and a negative objective value represents a profit. The computational results presented here are the first results requiring strict weekly frequency using a restricted version of the reference model to solve *LINER-LIB 2012*.

Instance	**Method**	**Best Cost**	**trans %**	**Depl. %**	**Time**
Baltic	Best weekly freq.	$-4.98 \cdot 10^6$	97.1	100	–
Baltic	Best weekly freq. + matheuristic	$-6.78 \cdot 10^6$	96.76	88.88	4.46
Baltic	MQKP+matheuristic (best)	$-8.60 \cdot 10^6$	97.23	88.89	2.05
Baltic	MQKP+matheuristic (Avg.)	$-1.19 \cdot 10^6$	94.77	88.89	4.17
Pacific	Best weekly freq.	$-6.64 \cdot 10^6$	96.80	98.10	–
Pacific	Best weekly freq. + matheuristic	$-3.57 \cdot 10^7$	96.70	93.5	623.45
Pacific	MQKP+matheuristic (best)	$1.03 \cdot 10^8$	96.02	94.39	825.96
Pacific	MQKP+matheuristic (Avg.)	$1.91 \cdot 10^8$	84.18	87.47	567.84

Table 5.6: Comparing the heuristic column generators best solution for the Baltic and Pacific case with an adjusted increased fleet to enforce strict weekly frequency. **Instance** denotes the instance. **Method** describes, which initial solution is used, and whether the matheuristic is invoked. **Best cost** denotes the best objective value found. When calculating with the construction heuristic as initial solution ten runs were performed and the best as well as the average solution is reported. **trans %** denotes the percentage of total cargo transported and **Depl. %** the percentage of the fleet deployed. **Time** denotes the CPU time in seconds.

In an attempt to compare the performance of the algorithms the best solutions from the Baltic case and the Pacific case have been altered to respect a weekly frequency by increasing the number of vessels in the fleet to meet this requirement. It is obvious, that the heuristic column generator could find different and possibly better solutions enforcing strict weekly frequency with the increased fleet. There will be excess capacity on the routes with biweekly frequency, which for the Baltic case is all vessel classes in the instance. In the Pacific case it represents half the vessel classes and less than half the total number of vessels (36%).

The Baltic case has been chosen as a reference because it is a small case and the matheuristic for LSNDP is relatively stable at finding good solutions for this instance, whereas the Pacific case is more complex and less influenced by biweekly frequency. The revenues and the fleet available, makes it hard to obtain a revenue and the matheuristic consistently performs poorly on this case.

The comparison takes the services from the best solution for the base case using the heuristic column generator adjusting the number of vessels to meet a weekly frequency requirement (Best weekly freq. in Table 5.6). They are used as initial solution and the matheuristic is able to improve upon the solution in its subsequent optimization for both instances. These results are compared against the same fleet using the construction heuristic. The matheuristic is able to find better solutions using the construction heuristic as opposed to the generated solutions for the Baltic case, but cannot compete on the Pacific case using the construction heuristic. In the Pacific case results show great variance in the revenue obtained and it is clear that this case is very dependant on the quality of the initial solution. The more complex instances have a solution space with many sub-neighborhoods and it seems that the matheuristic gets trapped in a local minimum, where it is not able to improve further or access more promising areas of the solution space. The results are presented in Table 5.6.

5.8 Algorithmic performance and results for *LINER-LIB 2012*

The results of testing the matheuristic on *LINER-LIB 2012* is presented in Table 5.8. For each instance ten replications with random seeds have been run. Randomness affects the initial solution constructed by the MQKP method and also the simulated annealing scheme incorporated in the subsequent improvement heuristic.

The algorithm has been set to terminate after the time limits imposed in Brouer et al. (2012) as stated in Table 5.7.

Baltic	WAF	Mediterranean	Pacific	AsiaEurope	WorldSmall
300	900	1200	3600	14400	10800

Table 5.7: Time limits imposed on each instance. the time is in CPU seconds.

The results are the first presented for a strict weekly frequency on rotations and the revenue is generally lower than the results presented in Brouer et al. (2012) as expected due to different frequency requirements. Results show a large deviation between the best found solutions and the average solution for most cases. The heuristic is very dependant on the quality of the initial solution, which deteriorates with instance size, as it is harder to get the right composition and fleet deployment as instance sizes grow. Furthermore, the solution space is complex with many sub-neighborhoods and the algorithm seems to get easily trapped in a local minimum. In the Mediterranean case the fleet deployment is very low and there is rejected cargo. It seems that the algorithm is not able to properly deploy the excess vessels to capture the residual demand. In general the fleet deployment is high, meaning that requiring a weekly frequency may not give sufficient capacity to transport the entire demand or the demands are not profitable to transport given the fleet /service configuration. The algorithm is fast for all instances but the AsiaEurope instance, which is the only one terminating due to the time limit imposed of 14400 seconds. The log files indicate that the loop structure over the entire set of S as illustrated in Algorithm 2 is in this case very time consuming and very little local search is performed within the time limit. The loop structures seem to be restraining at least for this case and it would be an enhancement to introduce an adaptive layer like the ones seen in ALNS algorithms (Pisinger and Ropke, 2007) to make more frequent and instance specific swaps between the neighborhoods and the local search procedure. Also a rating of the services to be optimized upon could be introduced as the MIP for some services return no moves for implementation in several contiguous for loops.

Since the solution times are generally very low it would be possible to implement a restart functionality with a new initial composition of services with a differing fleet deployment and seeding, perhaps based on the best found solution. Additionally, it is possible to experiment with additional neighborhoods and a more advanced local search procedure, than the one implemented. Additional neighborhoods could target swapping vessel classes between rotations and perform optimization of the port call sequence given the cargo allocated to the service. Speed optimization could also be enhanced. A more advanced local search could be used, where several methods for generating new services could be tested as this is a drawback of the current search especially for the larger instances. Generating routings of very high quality as seen in Brouer et al. (2012) could be implemented as there seems to be time for performing more advanced techniques for generating new promising routings.

5.8.1 Alternative method for generating new services

The local search implemented is relatively simple as described in Section 5.3.7. The services are rated based on the average utilization percentage of all individual voyages on the service. The resulting solution of removing the lowest utilized services is evaluated and at most two services are removed in each iteration. Subsequently new services are added based on the residual demand of the solution with the services removed. The original method creates a single service for each vessel

Instance			Cost(180 days)	Cost(weekly)	imp%	dep%	trans%	Time
Baltic	Low	Best	$3.11 \cdot 10^6$	124419	95.02	100.00	84.83	10.12
		Average	$1.19 \cdot 10^7$	476568	80.24	96.00	79.23	8.57
	Base	Best	$-5.90 \cdot 10^6$	-236038	110.21	100.00	92.07	3.65
		Average	$-7.32 \cdot 10^5$	-29261	101.66	100.00	89.92	4.06
	High	Best	$-5.87 \cdot 10^6$	-234718	107.52	100.00	92.84	9.14
		Average	$1.78 \cdot 10^6$	71144	98.15	96.10	88.21	11.76
WAF	Low	Best	$-9.39 \cdot 10^7$	$-3.76 \cdot 10^6$	221.03	100.00	90.15	13.22
		Average	$-8.24 \cdot 10^7$	$-3.30 \cdot 10^6$	198.82	96.67	84.84	10.89
	Base	Best	$-1.28 \cdot 10^8$	$-5.13 \cdot 10^6$	274.32	90.47	94.88	9.24
		Average	$-1.14 \cdot 10^8$	$-4.56 \cdot 10^6$	254.32	96.43	94.15	8.14
	High	Best	$-1.31 \cdot 10^8$	$-5.25 \cdot 10^6$	412.35	98.04	97.35	13.09
		Average	$-1.02 \cdot 10^8$	$-4.06 \cdot 10^6$	259.79	82.35	89.90	12.63
Mediterranean	Low	Best	$6.05 \cdot 10^7$	$2.42 \cdot 10^6$	42.67	78.26	81.20	155.78
		Average	$6.82 \cdot 10^7$	$2.73 \cdot 10^6$	37.85	86.52	80.98	98.56
	Base	Best	$4.09 \cdot 10^7$	$1.64 \cdot 10^6$	65.65	85.71	91.00	70.37
		Average	$5.36 \cdot 10^7$	$2.14 \cdot 10^6$	49.16	80.00	84.87	77.26
	High	Best	$4.10 \cdot 10^7$	$1.64 \cdot 10^6$	54.18	81.82	94.35	56.57
		Average	$4.78 \cdot 10^7$	$1.91 \cdot 10^6$	59.29	73.94	85.32	91.95
Pacific	Low	Best	$2.10 \cdot 10^8$	$8.40 \cdot 10^6$	70.23	97.56	81.14	359.76
		Average	$2.92 \cdot 10^8$	$1.17 \cdot 10^7$	61.20	92.56	79.29	368.39
	Base	Best	$4.36 \cdot 10^7$	$1.74 \cdot 10^6$	90.65	100.00	93.44	651.85
		Average	$1.62 \cdot 10^8$	$6.47 \cdot 10^6$	78.74	89.00	85.03	512.95
	High	Best	$-4.26 \cdot 10^7$	$-1.71 \cdot 10^6$	109.41	97.48	96.80	833.32
		Average	$1.20 \cdot 10^7$	$4.80 \cdot 10^5$	96.92	90.92	93.20	838.74
WorldSmall	Low	Best	$-4.76 \cdot 10^7$	$-1.90 \cdot 10^6$	101.98	99.02	83.63	5481.76
		Average	$6.93 \cdot 10^7$	2772230	93.09	97.30	79.84	5836.64
	Base	Best	$-7.97 \cdot 10^8$	$-3.19 \cdot 10^7$	178.36	95.82	91.83	8738.23
		Average	$-4.39 \cdot 10^8$	$-1.76 \cdot 10^7$	148.34	93.69	86.44	7881.96
	High	Best	$-1.05 \cdot 10^9$	$-4.18 \cdot 10^7$	433.05	94.64	92.89	4561.85
		Average	$-5.77 \cdot 10^8$	$-2.31 \cdot 10^7$	215.90	95.80	90.09	6735.87
AsiaEurope	Low	Best	$-4.45 \cdot 10^7$	$-1.78 \cdot 10^6$	103.60	100.00	83.17	14405.00
		Average	$7.04 \cdot 10^7$	2815218	94.41	98.14	81.07	13962.74
	Base	Best	$-3.07 \cdot 10^8$	$-1.23 \cdot 10^7$	151.10	98.30	93.18	14426.60
		Average	$-9.22 \cdot 10^7$	$-3.69 \cdot 10^6$	112.17	95.17	86.42	14502.02
	High	Best	$-5.03 \cdot 10^8$	$-2.01 \cdot 10^7$	185.96	96.70	95.70	14569.40
		Average	$-1.32 \cdot 10^8$	5288480	120.18	96.42	90.77	13836.17

Table 5.8: Overview of the results obtained for the benchmark suite *LINER-LIB 2012* using the matheuristic for LSNDP. The tests are based on ten runs with different random seeds. **Instance** is the name of the instance, **Best** is the best solution found, **Average** is the average of all ten runs. **Low** is the low capacitated case, **Base** is the medium capacitated case and **High** is the high capacity case. **Cost(180 days)** is the objective value for a planning horizon of 180 days similar to Brouer et al. (2012), whereas **Cost(weekly)** is the weekly cost/revenue of the network. Note that a negative value represents a profitable network. **imp%** is the percentage improvement from the objective value of the initial solution constructed by the MQKP construction heuristic, **dep%** is the percentage of the fleet deployed in terms of number of vessels, **trans%** is the percentage of demand transported and **Time** is the CPU time in seconds.

class targeting the largest residual demand in terms of the volume multiplied with the revenue per unit. Feasibility of the generated service given the duration of transporting the demand and the available number of vessels in the vessel class is considered in the method. The method is denoted *One Service Single Demand (OSSD)*. The approach creates very crude new services and in particular for the large instances with extensive use of transhipments the new services are

often removed in the subsequent local search as they have not evolved into a promising service. An alternative method (*MQKP*) invokes the construction heuristic for the ports present in the residual demand and the available fleet. We have tested the new method for the *Baltic* and the *WAF* instance. Ten runs of each instance was performed with identical seeds, such that the constructed solutions that are basis for the optimization are identical for both methods. Table 5.9 shows the average and best solutions found by each method. It can be seen that the more advanced method constructing new services using the construction heuristic gives better results for both instances on average. This means that introducing the MQKP method for larger instances should be investigated as it might improve the search.

Instance	Method	Best cost (weekly)	Average cost
Baltic	OSSD	-58165	245667
Baltic	MQKP	-41209	58601
WAF	OSSD	-4468130	-3843604
WAF	MQKP	-5128090	-4560574

Table 5.9: Average and Best weekly cost for two different methods of constructing a new service set in the local search. The ten runs are performed with the same random seed such that the basis for the optimization (the constructed solution)is identical for both methods.

5.9 The matheuristic as a decision support tool

The matheuristic can be used as a decision support tool for evaluating and altering a given network. A given network can be used as initial solution and it is furthermore possible to assign a subset of the services for optimization, while maintaining full knowledge of the capacity and flow in the remaining network. The matheuristic will keep the remaining network fixed and only apply the MIP neighborhood and local search to the designated subset of the services, while maintaining full knowledge of the entire network. Network planners are often responsible for a particular trade or geographic area and the method enables optimization of this designated part of the network. The method has been tested on a single case from Maersk Line.

The case is constructed from a real network at Maersk Line. The original network satisfies several constraints that are not incorporated in the matheuristic in its current form, such as several capacity types (DRY, REEFER, High Cube etc.), transit time restrictions, cabotage restrictions, and empty repositioning. Furthermore, the original network uses a larger set of vessel classes than the constructed case. The original network has been modified to fit the input of the matheuristic, which means that the capacity allocation has been set to match 6 vessel classes and demands have been mapped to a single capacity type without transit time restrictions, cabotage rules and empty repositioning. This means that the resulting case will be slightly overcapacitated and fulfills several constraints not considered by the matheuristic. This makes the mapped case an "easy optimization problem" compared to the real life case. Revenues have been constructed as an upper bound of the revenues found in the WorldLarge instance from *LINER-LIB 2012* and therefore, the objective value of the optimization does not represent the real value of the network in any way. Bearing these assumptions in mind, three cases have been constructed based on the mapped case optimizing on 3, 8 and 10 services respectively. These tests were performed on a regular desktop with an *Intel(R) Core(TM)2 Duo CPU P9400* at *2.40GHz* with 4 GB RAM.

The solutions calculated by the matheuristic are encouraging as there is a decent improvement in the cost, which for all cases show a revenue. The improvement seems to be primarily due to removing excess capacity looking at the deployment percentage (**Depl.%**). As a proof-of-concept implementation it shows that the matheuristic is capable of optimizing upon a real sized network with good results. It would be interesting to investigate an implementation fulfilling additional real-life constraints and testing it on a larger variety of cases and scenarios. Possible scenarios are adjustments to changes in demands and/or revenues, alternative fleet deployment and the introduction of new/changed services, that may have effects in other parts of the network.

Case	$\|S\|$	trans%	Depl.%	Cost	Time
Base Case with no optimization	0/139	99.5	100.00	$-5.67 \cdot 10^8$	140.59
Base Case - 3 services optimized	3/139	98.3	98.70	$-5.73 \cdot 10^8$	1862.48
Base Case - 8 services optimized	8/139	99.3	94.48	$-6.09 \cdot 10^8$	2385.66
Base Case - 10 services optimized	10/139	99.3	90.80	$-6.28 \cdot 10^8$	2803.58

Table 5.10: Tests optimizing a partial network based on Maersk Line network. $|S|$ denotes the number of services optimized upon compared to the full number of services in the network. **trans%** is the percentage transported of the total demand in units, **Depl.%** denotes the number of vessels deployed and **Cost** is the objective value (Again we are minimizing and a negative number means a profit). **Time** denotes the CPU time in seconds.

5.10 Conclusions

In this paper a matheuristic for the LSNDP has been presented using the reference model from Brouer et al. (2012) with strict weekly frequencies for all services. The matheuristic consists of a novel construction heuristic viewing the liner shipping network design problem as a specialized variant of the multiple quadratic knapsack problem. The initial solution is then improved using an integer program as neighborhood for each individual service considering insertion and removal of port calls based on estimation functions. The estimation functions allows us to circumvent solving a large scale multicommodity flow problem for evaluating a given candidate solution and may be applied in other heuristics for liner shipping network design. To circumvent the multicommodity flow a delta column generator is used to evaluate a candidate solution. The idea is to reuse the optimal basis for all variables not invalidated by the implemented moves and restart delayed column generation from here. A speedup of a factor 2-10 is achieved on average on the testbed for the *LINER-LIB 2012*. The improvement heuristic is used in combination with a simple local search based on a ruin-and-recreate principle, where vessels are freed by removing low utilized services and the resulting excess vessels are redeployed to services targeting the residual demand of a solution. Computational results are provided for the benchmark suite *LINER-LIB 2012* and are the first results enforcing strict weekly frequency. For twelve out of eighteen instances the algorithm seems to be performing well making a reasonable profit. For two cases (five out of six instances) the solutions are loss giving. Both cases have low or no profit using bi-weekly frequencies in Brouer et al. (2012) indicating that the solution spaces of the test cases do not contain many if any profitable solutions. The pacific case is profitable for bi-weekly frequency as seen in Brouer et al. (2012). The average performance is poor compared to the best solutions found, which indicates that the solution space is complex with many sub-neighborhoods and the matheuristic gets trapped in a local minimum too easily. The heuristic is also highly dependent on the initial solution and it may be worthwhile spending more time creating a good initial network by enhancing the construction heuristic. Finally, the matheuristic has been tested as a decision support tool with an existing liner shipping network as input only optimizing on a subset of the services in the network. Three cases are have been evaluated based on a real-life inspired network. The real life network contains additional constraints on the flow and hence is conceived as suboptimal for the matheuristic as these constraints are not enforced in the optimization. As a results the solutions from the optimized case may not be feasible in a real life network. Nevertheless, results are very encouraging as the algorithm is able to improve on all three cases tested and may prove as a valuable decision support tool for making incremental changes to a network and analyze different scenarios during network configuration.

Acknowledgements:

This project was supported by The Danish Strategical Research Council under the ENERPLAN project. We would like to thank the optimization team at Maersk Line for valuable input into optimization on liner shipping network design and their commitment to testing the matheuristic as a decision support tool. A special thanks to Joël Raucq for creating the base case and assisting

with testing the matheuristic as a decision support tool. His commitment and interest in the project has been very encouraging.

Bibliography

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

J. F. Alvarez. Joint routing and deployment of a fleet of container vessels. *Maritime Economics & Logistics*, 11(2):186, 2009.

C. Archetti, N. Bianchessi, A. Hertz, and M.G. Speranza. The split delivery capacitated team orienteering problem. Technical Report 55, GERAD, 2010. Les Cahiers du GERAD.

B.D. Brouer, J.F. Alvarez, C.E.M Plum, D. Pisinger, and M.M. Sigurd. A base integer programming model and benchmark suite for liner shipping network design. *Conditionally accepted at Transportation Science*, 2012.

S. Chen, B. Golden, and E. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4):318–329, 2007.

M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Barnhart and G Laporte, editors, *Handbooks in Operations Research and Management Sciences*, pages 189–284. North Holland, 14 edition, 2007.

K. Fagerholt. Designing optimal routes in a liner shipping problem. *Maritime Policy & Management*, 31(4):259 – 268, 2004.

K. Fagerholt, G. Laporte, and I. Norstad. Reducing fuel emissions by optimizing speed on shipping routes. *The Journal of the Operational Research Society*, 61(3):523–529, 2009.

R. De Franceschi, M. Fischetti, and P. Toth. A new ilp-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2):471–499, 2006.

D. Gulczynski, B. Golden, and E. Wasil. The split delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review*, 46(5): 612–626, 2010.

D. Gulczynski, B. Golden, and E. Wasil. The period vehicle routing problem: New heuristics and real-world variants. *Transportation Research Part E: Logistics and Transportation Review*, 47 (5):648–668, 2011.

M.G. Karlaftis, K. Kepaptsoglou, and E. Sambracos. Containership routing with time deadlines and simultaneous deliveries and pick-ups. *Transportation Research Part E*, 45:210–221, 2009.

K.H. Kjeldsen. Classification of ship routing and scheduling problems in liner shipping. *INFOR - special issue on maritime transportation*, 49(2):139–152, 2011.

V. Maniezzo, T. Stützle, and S. Voss. *Matheuristics: hybridizing metaheuristics and mathematical programming*. Springer, New York etc., 2009.

Q. Meng and S. Wang. Liner shipping service network design with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, 47(5):695–708, 2011.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403–2435, 2007.

C.E.M. Plum. The linearized simultaneous string-design and cargo-routing problem. EURO 2010 conference, 2010.

K. Rana and R.G. Vickson. Routing container ships using lagrangian relaxation and decomposition. *Transportation Science*, 25(3):201–214, 1991.

L. B. Reinhardt and D. Pisinger. A branch and cut algorithm for the container shipping network design problem. *Flexible Services and Manufacturing Journal*, pages 1–26, 2011. Article in Press.

M. Stopford. *Maritime Economics*. Routledge, third edition, 2009. ISBN 0203442660.

WSC. The liner shipping industry and carbon emissions policy, 2009. URL `www.shippingandco2.org/LinerShippingandCO2EmissionsPolicySeptember.pdf`. The World Shipping Council (WSC) - Last viewing September 2011.

WSC. How liner shipping works, 2011. URL `http://www.worldshipping.org/about-the-industry/how-liner-sh` The World Shipping Council (WSC) - Last viewing September 2011.