The time constrained multi-commodity network flow problem and its application to liner shipping network design

Karsten, Christian Vad; Pisinger, David; Røpke, Stefan; Brouer, Berit Dangaard

Published in:
Transportation Research. Part E: Logistics and Transportation Review

Link to article, DOI:
10.1016/j.tre.2015.01.005

Publication date:
2015

Document Version
Peer reviewed version

The time constrained multi-commodity network flow problem and its application to liner shipping network design

Christian Vad Karsten, David Pisinger,
Stefan Ropke and Berit Dangaard Brouer
Department of Management Engineering,
The Technical University of Denmark

January 13, 2015

Abstract

The multi-commodity network flow problem is an important sub-problem in several heuristics and exact methods for designing route networks for container ships. The sub-problem decides how cargoes should be transported through the network provided by shipping routes. This paper studies the multi-commodity network flow problem with transit time constraints which puts limits on the duration of the transit of the commodities through the network. It is shown that for the particular application it does not increase the solution time to include the transit time constraints and that including the transit time is essential to offer customers a competitive product.

1 Introduction

According to IMO (2014) 90% of global trade is carried out via the sea, and ships flying EU flags emit more than 20 million tons of CO_2 (MaritimeCO2, 2014). Container Shipping involves the transportation of a major share of the worlds goods and has been steadily growing (with a small decrease around 2009 due to the economic crisis). Reliance on container shipping to transport goods internationally is only expected to increase due to its economic advantages compared to other transportation modes. Additionally, the CO_2 emissions per ton cargo transported using maritime transport is significantly lower than road and rail transport. Hence, even small improvements in the underlying network of a liner shipping company can have a significant impact, both economically and environmentally. Despite this, the *Liner Shipping Network Design* (LSND) problem has not received a lot of attention in the Operations Research literature and it is far from being a well-solved problem (Meng et al., 2014). Christiansen et al. (2004) and Christiansen et al. (2013) provide comprehensive reviews of the literature published within the field of maritime optimization and liner shipping.

A liner shipping network consists of a number of rotations, which are round trips. It is common to have weekly departures at each port, hence a sufficient

number of vessels are deployed to each rotation, to ensure the requested frequency. Figure 1 shows an example of a real-world rotation. Different vessels have varying capacity and speed, and the transport of a commodity through the network may include the use of several rotations to connect between the origin and destination port. The switch from one rotation to another is referred to as *transshipment* and there is a cost associated with this since the container must be handled by the quay-cranes at the transshipment ports and possibly stored temporarily at the container yard. On top of this, the container will experience a wait time during the transfer process. Because of the associated cost and transit time and the risk of goods being damaged containers are at most subject to a few transshipments, when traveling from their origin to destination. The *transit time* is the time it takes a commodity to travel from origin to destination. Transit time is counted in days and allowed transit times may vary from one day to several months Brouer et al. (2014). Liner shipping networks that are optimized only with respect to cost get an unrealistically high network utilization as containers are allowed on detours that offer unused capacity but in practice they will violate transit time restrictions.

Given a candidate network a multi-commodity network flow (MCF) problem is solved in order to decide, which of the available cargoes should be shipped on which routes. An extensive treatment of the MCF problem can be found in e.g., Ahuja et al. (1993). The MCF problem can be formulated as a linear programming problem which can be solved in polynomial time and there are many algorithms for solving it. One of these methods is by delayed column generation, see Desaulniers et al. (2005) and Ahuja et al. (1993). In order to include the transit time constraint one has to solve an extended version of the problem, the time constrained MCF problem, which is NP-hard. This is easily shown by reduction from the shortest weight constrained path problem, (Garey and Johnson, 1979).(We transform this problem into a time constrained MCF by having only one commodity with source and destination as given by the shortest path problem.) This paper presents an algorithm for the time constrained MCF problem and given the LSND application several possible improvements are presented.

To the best of our knowledge, most algorithms for the LSND problem do not include transit time restrictions for shipped commodities. This paper studies the consequence of neglecting the transit time restrictions in existing networks. This is done by taking the networks produced by a LSND heuristic and comparing the estimated revenue with and without including the time constraint in the cargo flow calculations. The results show a substantial difference, and we therefore recommend that future LSND algorithms should include the transit time constraint if possible.

The remainder of the paper is organized as follows. In Section 2 we discuss the *level of service* in liner shipping and review relevant literature. In Section 3 we introduce the multi-commodity flow problem with time constraints and describe a delayed column generation procedure for solving it. Furthermore, we discuss a way to tailor the resource constrained shortest path problem, which arise as the sub-problem in the column generation process, in order to solve it efficiently. Section 4 describes a contraction scheme for the graph, which reduces the number of edges in certain instances of the graph to speed up the sub-problem computations. Section 5 introduces novel ways of modeling the

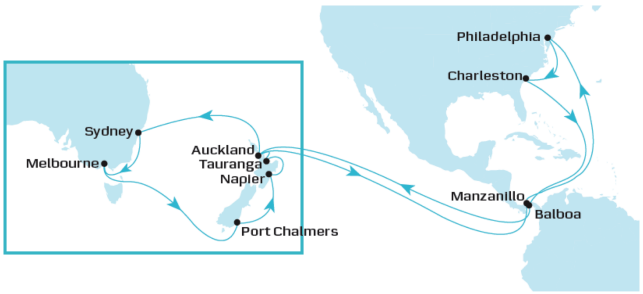

Figure 1: An example of a sailing route (rotation) in the Maersk Line network, from Maersk (2014).

transshipments to accommodate different network design model scopes. Finally, we conduct computational experiments in Section 6 and investigate the sensitivity of the travel time restrictions.

2 The Level of Service in Liner Shipping

Several factors such as price, transit time, transshipments, port coverage, frequency, reliability, administration, equipment, environmental friendliness and schedules can be relevant and important for a shipper when considering different carriers, (Brouer et al., 2014). Hence it is important to meet these constraints when constructing and evaluating liner shipping networks. The cost and transit time are often identified as the most important factors, (Meng et al., 2014; Brouer et al., 2014; Gelareh et al., 2010; Notteboom and Vernimmen, 2009; Notteboom, 2006), however, most previous work within LSND neglects transit time and mainly considers cost.

Designing networks with focus only on cost has the apparently attractive benefit that reducing cost goes hand in hand with reducing CO_2 emissions as fuel is the largest cost component. Reducing CO_2 emissions is an important goal of several governments, and it is generally attractive for carriers as well as shippers to have a green profile. Slow steaming is one common way of both reducing cost and emissions, but this requires a broader introduction of the level of service requirements in the network design models. There is an inherent trade off between reducing bunker consumption and thereby emissions through speed reduction and offering competitive transit times for commodities.

On the other hand, by offering a time competitive mode of transport more cargo will be transported this way, reducing the global CO_2 emissions. By introducing a maximum transit time for each commodity in the network, the number of allowed paths will be limited significantly for the individual commodities and introduces new limits on the feasible solutions in the network design process. However, it requires adding a time dimension to all edges in a network and especially the service time at ports and the time spent transshipping between rotations need careful analysis to obtain both competitive network cost and transit times. In order for a network to be competitive it must offer low transit times and few transshipments.

Implications of travel time restrictions is not well-studied in connection with LSND, but recently it has been studied in connection with related problems.

Agarwal and Ergun (2008) present a time-space graph to introduce a rough schedule of weekdays in the network design process, but they do not introduce travel time restrictions and do not account for the cost of transshipping goods. Gelareh et al. (2010) study a hub-and-spoke network design problem for two liner shipping companies in a competitive environment. The market share is determined by transit time and transportation cost. Wang and Meng (2011) study schedule design and container-routing for a given network with predefined paths. They minimize the transshipment cost, add a penalty cost for longer transit times and a bonus for shorter transit times. Wang and Meng (2012) give a tactical model for schedule design, where they minimize the cost, while maintaining a required transit time taking time uncertainty into account. Meng and Wang (2012) study the fleet deployment problem in conjunction with transit time levels in a space–time network. Wang et al. (2013) study an integer program for generating a container path for a single OD-pair taking transit time and cabotage rules into account. A case study considering a single path is presented. No computational run time is reported. Plum et al. (2014) consider transit time for the design of a single rotation with up to 25 ports. Finally, Wang and Meng (2014) present a non-linear mixed integer model for the network design problem taking transit time into account and formulate a column generation based heuristic for solving it for a Europe Asia network with 12 ports. Álvarez (2011) gives mathematical expressions for the transit time of goods, which is composed of time at sea, time at ports and dwell time, and derive a bi-linear cost expression for the inventory holding.

Neither exact nor heuristic solution methods are yet able to solve LSND instances with the size of a global carrier to (near) optimality, but a promising approach is to rely on a two-tier structure as in Álvarez (2011); Brouer and Desaulniers (2012); Brouer et al. (2014), where route planing, fleet deployment and sailing speed is determined in the upper tier corresponding to determining the cost of the network, while the lower tier determines the revenue of the network by flowing the available cargo. In the following, we consider the cargo flow subproblem, which is one of the main challenges in LSND. In Brouer et al. (2011) a specialized MCF considering liner shipping cargo flow with empty repositioning is presented along with a computational study of solving the LP arc-flow model versus solving a path-flow model using column generation. Holmberg and Yuan (2003) discuss general MCF problems with side constraints and propose a column generation procedure for solving them, the solution method in this paper is similar to that of Holmberg and Yuan (2003), but specialized to the LSND application.

It is worth mentioning that graph representations and commodity flows within the maritime area are studied outside the core Operations Research community. Examples are Kaluza et al. (2010), Ducruet and Notteboom (2012) and Ducruet (2013) who create aggregate graphs representing vessel movements by combining the historic trajectories of individual vessels. The papers analyse the aggregate graphs, for example with respect to change over time (Ducruet and Notteboom, 2012) or with respect to the importance of diversification of port activities (Ducruet, 2013).

3 Time-constrained Multi-commodity Flows

As mentioned above, a promising approach for solving the LSND problem heuristically is to use a two phase approach. The first phase builds a network consisting of a number of rotations and the second phase decides, how cargo should be transported in this network to evaluate the cost/revenue of the network. In this paper we do not consider network design, instead we focus solely on determining how cargo should flow through the network.

Figure 2a) illustrates a basic network that is the output of Phase 1. In this example the network is composed of two rotations $R1$ and $R2$. In general the graph contains the node set N as consisting of P and C, ports and calls respectively. Goods can be transshipped between rotations at the port, where rotations meet. I.e. goods can be transshipped between rotation $R1$ and $R2$ in node B. The flow of goods through the network is decided in the second phase. This is illustrated in Figures 2b) and 2c). In this example we only have three commodities. Ten units of commodity $K1$ is based in node A and destined for node C, ten units of $K2$ is based in A and destined for B, and ten units of $K3$ is based in B and destined for C. Transporting one unit of commodity $K1$, $K2$, and $K3$ results in an income of 10, 4, and 4, respectively. The capacities of the edges in the network is determined by the capacities of the vessel class used and the frequency of the rotation. In this example we assume that all the edges have a capacity of 10. When solving the cargo flow problem no cost is associated with traversing the voyage edges, as we assume the sailing cost to be roughly identical whether or not the ship is fully loaded, an assumption that is not completely true in practice [1]. The cost of operating the ships will be accounted for in Phase 1. In the cargo flow phase we do pay for each transshipment action and loading and unloading. In this example, we assume, that the cost is one per unit transshipped and neglect load/unload costs.

Figure 2b) shows the optimal solution to the cargo flow problem in our example. We can only transport a total of 10 units through the two rotations $R1$ and $R2$ and hence, transporting 10 units of commodity $K1$ gives the highest revenue (90). Now consider that the traversal of each edge and each transshipment action takes one time unit and consider that all commodities must reach their destination within 2 time units. Therefore, the solution found without transit time restrictions, is no longer feasible. The optimal solution given the time restriction is shown in Figure 2c). Here it is possible to ship commodity $K2$ and $K3$. The resulting revenue is 80 since we do not have to pay for the transshipment operation.

In the following we first review the MCF problem and later show how a time-constrained MCF can be modeled and solved.

The arc flow formulation MCF problem can be stated as follows. We redefine $G = (N, A)$ to be a generic, directed graph with nodes N and edges A. Let K be the set of commodities to transport and b^k be the amount of commodity $k \in K$ that is available for transport. We assume that each commodity has a single origin node and a single destination node denoted $o(k)$ and $d(k)$, respectively. Let u_{ij} be the capacity of edge (i, j). For each node $i \in N$ and commodity

[1] In reality a typical container ship uses more fuel when traveling fully loaded compared to sailing empty.

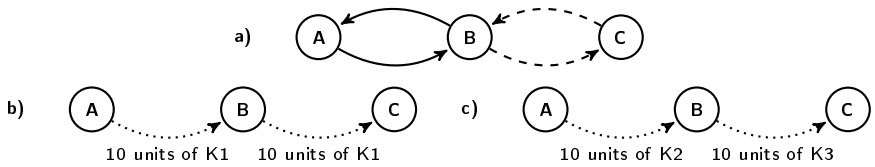

Figure 2: a) A simple example network with two rotations $R1$ (solid edges) and $R2$ (dashed edges). b) and c) show two possible flows in the network. The paths for commodities K1, K2, and K3 are marked by dotted edges.

$k \in K$ we define

$$b(i,k) = \begin{cases} b^k & \text{if } i = o(k) \\ -b^k & \text{if } i = d(k) \\ 0 & \text{otherwise} \end{cases}$$

and for each node $i \in N$ we define the sets $\delta^+(i) = \{(j,j') \in A : j = i\}$ and $\delta^-(i) = \{(j,j') \in A : j' = i\}$, that is, the set of edges with tail and head in node i, respectively. The model uses decision variables x_{ij}^k that specify the amount of commodity $k \in K$ that flows through edge (i,j). We do not impose integrality conditions on the flow as in practice several thousand containers are moved on a single vessel and hence fractional containers are negligible. Additionally the demand is often a forecast so the variation in this will exceed rounding errors. Brouer et al. (2011) investigate the effect of integrality for the version of the problem without time-constraints and find that most solutions are integral in practice and that the gap in terms of objective value for the considered real-life instances never exceeds 0.01% if a fractional solution is just rounded. For each unit of commodity k that flows through edge (i,j) the cost is c_{ij}^k. With this notation the MCF problem can be stated as a linear programming problem as follows

$$\min \sum_{(i,j)\in A} \sum_{k \in K} c_{ij}^k x_{ij}^k \tag{1}$$

subject to

$$\sum_{(j,j')\in\delta^+(i)} x_{jj'}^k - \sum_{(j,j')\in\delta^-(i)} x_{jj'}^k = b(i,k) \quad \forall i \in N, k \in K \tag{2}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} \qquad \forall (i,j) \in A \tag{3}$$

$$x_{ij}^k \geq 0 \qquad \forall (i,j) \in A, k \in K \tag{4}$$

The objective function (1) minimizes the cost of the chosen flow, constraint (2) ensures flow conservation and ensures that commodities originates and terminates in the right nodes. Constraint (3) ensures that the capacity of each edge is respected. This formulation has $|K||A|$ variables and $|A| + |K||N|$ constraints. The number of variables is hence polynomially bounded, but for large graphs like the ones seen in global liner shipping networks this formulation requires excessive computation time and may even be too large for standard linear programming solvers (see e.g. Brouer et al. (2011)).

It is not hard to see how the MCF can be used to find the optimal cargo flow in the LSND given a set of rotations, but we would like to make a comment on transshipments and rejected demands. The most straightforward approach for modeling transshipments is to model each transshipment port by a node for each rotation that visits the port. Edges between nodes from different rotations, meeting at a transshipment ports, are used to model the actual transshipment. The cost of such edges is equal to the cost of the transshipment. Section 5 discusses this and other modeling approaches. The standard MCF model written above enforces that all demands are being met. We can let the model reject demand by including dummy arcs between source and destination with an appropriate penalty.

An alternative model for the MCF is the path-flow formulation where each variable corresponds to a path through the graph for a certain commodity. To define the model we need to define the following sets: let Ω^k be the set of all feasible paths for commodity k, $\Omega^k(a)$ be the set of paths for commodity k that uses edge a and $\Omega(a) = \cup_{k \in K} \Omega^k(a)$ is the set of all paths that use edge a. We have a variable x_j for each path j. The variable states, how many units of a specific commodity that is routed through the given path, the cost of each variable is given by the parameter c_j. The model is

$$\min \quad \sum_{k \in K} \sum_{j \in \Omega^k} c_j x_j \tag{5}$$

$$s.t. \quad \sum_{j \in \Omega^k} x_j = b^k \qquad \forall k \in K \tag{6}$$

$$\sum_{j \in \Omega(a)} x_j \leq u_{ij} \qquad \forall (i,j) \in A \tag{7}$$

$$x_j \geq 0 \qquad \forall k \in K,\ j \in \Omega^k \tag{8}$$

Here constraint (6) ensures that the demand of each commodity is met and constraint (7) ensures that the capacity limit of each edge is obeyed. The path-flow model has |A|+|K| constraints, but the number of variables is, in general, growing exponentially with the size of the graph. However, using delayed column generation the necessary variables can be generated dynamically and in practice the path-flow model can often be solved faster than the arc-flow model for large scale instances of the LSND problem (see Brouer et al. (2011)).

Delayed column generation works with a reduced version of the LP (5)-(8), which is called the master problem. The master problem is defined by a reduced set of columns $\bar{\Omega}^k$ for each commodity k such that a feasible solution to the LP (5)-(8) can be found using variables from $\cup_{k \in K} \bar{\Omega}^k$ (if there is no available connection a forfeited edge with a penalty cost is used.). Solving this LP gives rise to dual variables π_k and λ_{ij} corresponding to constraint (6) and (7), respectively. For a variable $j \in \cup_{k \in K} \Omega^k$ we let $\kappa(j)$ denote the commodity that a variable serves and let $p(j)$ represent the path corresponding to the variable j, represented as the set of edges traversed by the path. Then we can calculate the *reduced cost* $\bar{c}_j$ of each variable $j \in \cup_{k \in K} \Omega^k$ as follows

$$\bar{c}_j = \sum_{(i,j) \in p(j)} (c_{ij}^{\kappa(j)} - \lambda_{ij}) - \pi_{\kappa(j)}.$$

If we can find a variable $j \in \cup_{k \in K} (\Omega^k \setminus \bar{\Omega}^k)$ such that $\bar{c}_j < 0$ then this variable

has the potential to improve the current LP solution and should be added to the master problem, which is resolved to give new dual values. If, on the other hand, we have that $\bar{c}_j \geq 0$ for all $j \in \cup_{k \in K}(\Omega^k \setminus \bar{\Omega}^k)$ then we know the master problem defined by $\bar{\Omega}^k$ provides the optimal solution to the complete problem (for more details see Alvarez (2009); Ahuja et al. (1993)). In order to find a variable with negative reduced cost or prove that no such variable exists we solve a sub-problem for each commodity. The sub-problem seeks the feasible path for commodity k with minimum reduced cost given the current dual values. It is not hard to see that solving this problem amounts to solving a shortest path problem from source to destination of the commodity with edge costs given by $c_{ij} - \lambda_{ij}$ and subtracting π_k from this cost in order to get the reduced cost. We note that $\lambda_{ij} \leq 0$, which means that the edge cost in the sub-problem will be non-negative.

We add a constraint on the transit time of the voyage of each commodity to accommodate the transit time restrictions. Adding this constraint to the arc-flow model is non-trivial since the demand for each commodity can be fulfilled using multiple paths. In this formulation multiple paths are bundled up in a tree structure, where the time of each individual path cannot easily be tracked. In the path-flow formulation the constraint can be handled in the definition of Ω^k, ensuring that the set only contains paths that are feasible with respect to the transit time constraint. The formulation separates each of the paths into a single variable, enabling us to track time of each individual commodity. However, doing so complicates the delayed column generation algorithm since the sub-problem has to ensure that the transit time of each generated path is less than or equal to the maximum transit time for the given commodity. This changes in this case the sub-problem from being an ordinary shortest path problem solved e.g. using Dijkstra's algorithm to a weakly NP-hard *resource constrained shortest path*, RCSP, problem, (Hassin, 1992).

Detailed network description

We define the set of voyage edges, A_v, as the set of edges connecting two nodes in C on the same rotation, i.e. consecutive port calls on a rotation and $A_v = \{(i,j)|i,j \in C \wedge i,j \in r'\}\}$. The time, t_a, to traverse arc $a \in A_v$ is calculated according to the distance sailed with the average speed of the rotation. An edge connecting two calls in the same port is denoted a transshipment edge belonging to the edge set $A_t = \{(i,j)|i,j \in C \wedge i \in r_1, j \in r_2\}$. $t_a \; for \; a \in A_t$ denotes the transshipment time and $c_a \; for \; a \in A_t$ the transshipment cost. As we do not have a schedule in the following we work with an average transshipment time of three days, i.e., $t_a = 3 \; for \; a \in A_t$. Every load and unload of a unit of cargo is associated with a cargo handling cost. Hence, for the set of (un)load edges, $A_l = \{(i,j)|(i \in C \wedge j \in P) \vee (j \in C \wedge i \in P)\}$, $t_a \; for \; a \in A_l$ denotes the handling time and $c_a \; for \; a \in A_l$ denotes the load/unload cost. We set the load and unload time to one day, i.e., $t_a = 1 \; \forall \; a \in A_l$. Lastly, it is possible to omit a cargo using the set of forfeited edges $A_f = \{(i,j)|(i,j \in P) \wedge (\exists k \in K, o(k) = i \wedge d(k) = j)\}$. $t_a, \; for \; a \in A_f$ denotes the maximum allowed transit time and c_a is a goodwill penalty for not transporting the cargo. We assume that the loading and unloading as well as transshipment times in a port are independent of the number of containers to be handled at the port. The edge set A is defined as $A = A_v \cup A_t \cup A_l \cup A_f$.

3.1 Resource Constrained Shortest Path Calculations

The RCSP sub-problem can be solved using various methods. One method is to use a label setting algorithm as proposed in Irnich and Desaulniers (2005). Labeling algorithms are based on dynamic programming and use resource extension functions and dominance functions to efficiently calculate the shortest path through a graph considering several resources, here (reduced) cost and time. The resources must be of a form where they can be determined at the vertices of a directed walk in a graph. We say that a resource is *constrained* if there is at least one vertex in the graph where the resource is bounded from above, otherwise the resource is *unconstrained*. We treat (reduced) cost as an unconstrained resource, which we minimize, and time as a constrained resource, as the limits on transit time, limits the time resource in the algorithm. When solving the MCF problem using the labeling algorithm the accumulated consumption of the resources is non-decreasing in each extension of a label. This is a prerequisite for the algorithm to work. Labels are used to store the information on the resource values for (incomplete) paths through the graph. Labels are associated with the vertices in the graph and they are propagated via resource extension functions along the edges in the graph. An extension of a label is feasible if the resulting label is feasible, i.e. the transit time did not exceed the limit. A decisive feature of the algorithm is to keep the number of labels as small as possible. This is done via a dominance function, which eliminates unnecessary labels. The dominance function checks if all resources, i.e. cost and time, for one label is less than or equal to the value of the resource in the other label at each vertex, i.e., a label, l_a, is dominated by another label, l_b, if $cost(l_a) \leq cost(l_b)$ and $time(l_a) \leq time(l_b)$. This improves the running time of the algorithm, since dominated labels need not to be extended and can be deleted. Pseudo code is given in Figure 3.

At each iteration, the labeling algorithm selects a label from the set of unprocessed labels U and checks it for dominance and feasibility. If the label is dominated it is deleted, whereas if it is undominated, it is extended along all out-edges of the current vertex. If the new label is also feasible it is added to the set of unprocessed labels and to the set of labels residing at the successor vertex. If the new label is not feasible, it is deleted. The algorithm stops, when there are no more unprocessed labels. Then it determines whether the destination vertex can be reached and constructs all undominated (Pareto-optimal) paths. Hence, tight limits on transit time in a large network will cause the algorithm to terminate faster as fewer labels need to be extended.

Reducing the number of SPP calculations

In the column generation procedure of the MCF problem a RCSP problem must be solved for each of the commodities with individual restrictions on travel time for all commodities. However, the natural origin-destination (o-d) implementation for each commodity suggested by the MCF problem can be modified. The RCSP algorithm is executed for the commodity with the maximum allowed transit time from the set of commodities with identical origin. As a "by product" the shortest paths for the remaining commodities with identical origin are also found. This is due to the nature of the label setting algorithm, where labels represent paths. All labels that are not dominated (reduced cost and time) and

Require: a graph, G, with corresponding node and edge descriptors
Require: a node descriptor, $s(k)$, for the start node of a path
Require: a set of node descriptors, E, containing destinations of demands with origin $s(k)$
Initialize Initialize the set of unprocessed label $U = \{s\}$
T=max(allowed transit time of all commodities leaving s)
while $\mathcal{U} \neq \emptyset$ **do** $current\_label \leftarrow \min(\mathcal{U})$
if $current\_label$ is not dominated **then**
node i = ResidentNode($current\_label$)
check dominance and delete dominated and processed labels
mark $current\_label$ as processed
for all outgoing edges, (i, j), of i **do**
new_label = resource_extension_function($current\_label$)
if $new\_label.time > T$ (i.e. not feasible) **then** delete new_label
else $U \leftarrow new\_label$
else delete $current\_label$
for all $e \in E$ **do** add paths, pe, and resource consumption for $e(k)$ to PE
for all $pe \in PE$ **do**
if transit time violate allowance for pe **then** delete path
else add path to set of feasible paths, FE

Figure 3: Pseudo Code, Resource Constrained Shortest Path, o-all

do not violate the travel time restriction for the commodity with the longest allowed are not deleted. Hence, we are guaranteed to find all optimal paths for the commodities with origin o if such exist. At the end of the algorithm all paths to a node are considered and a post processing procedure that erases paths violating the allowed transit time for each commodity is implemented, see the pseudo code in Figure 3. Hence, at most the number of ports $|P|$ RCSP calculations are needed to obtain o-d paths for all commodities, but still it is possible to use the domination. If using the o-d implementation of the algorithm we would need $|K|$ calculations. For a global network the number of ports is significantly less than the number of commodities $|P| < |K|$. In the WorldSmall instance provided in Brouer et al. (2014) there are $|P| = 47$ ports and $|K| = 1764$ commodities and in the AsiaEurope instance there are $|P| = 111$ ports and $|K| = 4000$ commodities.
The algorithm is based on a variation of the The Boost Graph Library (BGL) implementation. It uses a resource extension function to specify extensions of labels, and a dominance function comparing cost and time for two labels.

4 Graph Contraction

The computational time increases with the size of the graph, but due to the inherent structure of the networks in Liner Shipping it is possible to simplify the corresponding graphs for each of the sub-problems. Figure 4a) shows a graph representation of the voyage edges in a small instance with five rotations $(B \to I \to J)$, $(C \to Y \to Z \to X)$, etc. All edges have a cost of one. There are four minor hubs B, C, F and N, where transshipments from one rotation

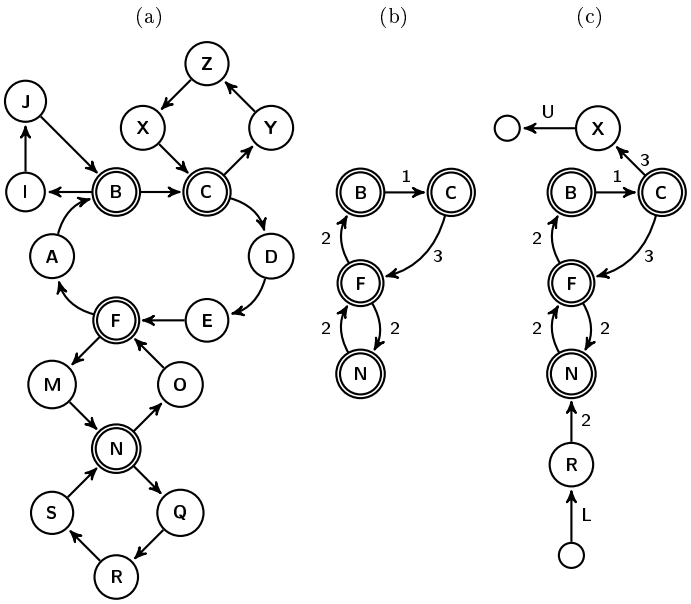

Figure 4: A graph representation of the voyage edges in a small instance with five rotations. All edges have a cost of one. There are four minor hubs B, C, F and N, where transshipments from one rotation to another are possible. Load, unload and transshipment edges have been excluded for simplicity.

to another are possible. We contract this graph to one where only hub nodes are kept and edges represent voyage possibilities between hubs. This graph is shown in Figure 4b). An edge in this graph is a contraction of one or more edges from the original graph. The edge $C \rightarrow F$ for example represents the path $C \rightarrow D \rightarrow E \rightarrow F$ in the original graph. The simplified graph does not contain all nodes from the original graph so many of the needed shortest path computations are not possible in the reduced graph. However, for each necessary shortest path computation we extend the graph as necessary. This is illustrated in Figure 4c). The figure shows the graph that is necessary to compute the shortest path from R to X. Nodes R and X are added to the graph. Node R connects to the contracted network through node N so an edge is added from R to N with the appropriate cost (the cost of $R \rightarrow S$ plus the cost of $S \rightarrow N$) and node X can only be reached through node C so an edge is added from C to X with appropriate cost. Also a load edge, L, is included to account for loading cost and time as well as an unload edge, U. Hence in contrast to Brouer and Desaulniers (2012) the graph only includes relevant load/unload edges. Figure 5 shows the pseudo code for the contraction algorithm. After the contraction of the graph, it is modified separately for each commodity or commodity group such that edges connecting the load port and the destination port(s) with the contracted network are added if these are not hubs. Likewise, load and unload edges are added.

As mentioned in Section 3.1 we prefer to do shortest path calculations with a single origin and many destinations. The multi-destination calculations are

Require: a graph, G, with edges and nodes corresponding to the transportation network and a copy, G', only containing the nodes
for all rotations, r, in G **do**
find degree of nodes in r to determine whether it is a transshipment node.
if # transshipment nodes > 1 **then** determine $first\_voyage\_edge$ on r
while next node $\neq$ first node **do** find next port and voyage edge on r
add current voyage edge info to update $current\_contracted\_edge$
if $degree\_destination\_node \leq 2$ **then** continue
else add $current\_contracted\_edge$ to G'
clear $current\_contracted\_edge$

Figure 5: Pseudo-code for contracting a graph

also possible in the contracted graphs by adding appropriate edges for each destination in the same way as described for a single destination shortest path calculation. The *Reduced Graph* decreases the number of extensions needed in the label setting algorithm for the shortest path calculations and hence speed up the computation. This approach is more tractable when only a few ports are hubs (i.e. visited by more than one rotation). We use the network structure presented in Brouer and Desaulniers (2012) as a reference, denoted the *Full Graph*. Additionally to reduce the size of the reduced graph used during the SPP calculations we only consider the load and unload edges which are relevant to the considered set of commodities as well as the relevant forfeited edges in contrast to Brouer and Desaulniers (2012).

5 Representation of Transshipments

There are several ways of handling transshipments in the graph. Each modeling approach has different properties and benefits. An alternative modeling approach to the ones presented in the following is given in Plum et al. (2013).

Figure 8-11 show different graph representations of a transshipment structure. In most cases a port node is augmented to contain internal port nodes and edges such that the cost, capacity, and time of the port operation can be correctly accounted for. We are going to analyse the structures in Figure 8-10 in further detail in the computational section while we just want to mention some additional properties of the structures shown in Figure 6-11.

The structure in Figure 6 is the most basic representation of a transshipment and it does not allow modeling of neither cost nor time related to transshipments as commodities transfer directly between the rotations. No additional nodes or edges are added and a commodity will transfer directly from one voyage edge to another. Figure 7 shows a generalized version of this simple structure where each port call is assigned a transshipment node and these are connected in a "ring". This requires r extra edges and r extra nodes, where r is the number of rotations visiting a hub. This allows modeling of a schedule and the time between two services (if ordered in terms of arrival) can be added to the edges. It is however not possible to correctly account for transshipment costs and buffer time.

The complete structure found in Figure 8 is used in e.g. Brouer and Desaulniers (2012), and may be the most intuitive representation of a transship-

ment as all rotations visiting a port are directly connected to all other rotations visiting the same port. This allows different costs and transit times between different rotations, which can be calculated directly according to some given schedule including buffer time. This comes at a cost of having a high number of edges in larger hubs. It requires $r(r-1)$ edges and r nodes. Each edge has an associated cost and time. The representations in Figure 9 and in Figure 10, denoted *star* and *ring* respectively, mitigate these costs by introducing one or several additional transshipment nodes. The number of edges only increases linearly with the number of rotations visiting a hub.

The ring structure, like the complete structure, allows individual transit times based on a given schedule, whereas the star structure does not. The star structure, which is also used in Wang and Meng (2013), introduces one new transshipment node, At, and has $2r$ edges and $r+1$ nodes. All rotations are connected to the transshipment node via an edge with an associated cost and time. Edges out of the transshipment node have no associated cost or time. The ring structure introduces a new transshipment node for each port, in the figure $At1$, $At2$, $At3$, and $At4$ respectively, and new edges in a "ring" with an associated time and cost. The edges leaving the transshipment nodes has no associated cost or time in our experiments but as discussed below adding cost or time allows modeling of additional properties. The structure has $3r$ edges and $2r$ nodes. Both the complete and ring structure makes it possible to consider an actual schedule with arrival and departure time specified, however it is not possible to take buffer time between two rotations into account in the ring structure. The star structure offers a simpler structure than the complete and ring structure if average transit times are considered and not actual schedules. Both the ring structure and the star structure can additionally handle operational capacities in the port such as quay crane capacity, i.e. adding a capacity to the edges between A and At will ensure that the number of containers loaded and unloaded to/from all services in the port does not exceed that capacity of the quay or the cranes assigned to a given vessel. This is not possible to model in the complete structure. In the cases, where the total travel time is close to the limit, it is sufficient to check the initial transshipment edge to cut off all possible transfers in the star and ring structure. Hence for the purpose of evaluating the algorithmic effects of considering cargo transit times we use the structures in Figure 8 - 10 in the computational experiments.

Finally, the structure in Figure 11 is a generalization of the discussed models where it is possible to take both port productivity in terms or crane and quay capacity as well as buffer time between two rotations into account. This structure requires $r(r-1)+2r$ edges and $2r$ nodes.

In practice, to reduce the number of edges further, we can combine the structures such that for all physical ports if the number of visiting rotations $>$ 3 (i.e. it is a hub with more than 3 visiting rotations) we change transshipment layout to either the star or ring structure, whereas for hubs with ≤ 3 rotations visiting we use the complete structure in all cases.

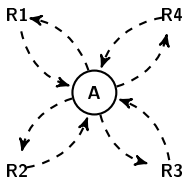

Figure 6: Simple transshipment structure shown for a physical port, A, with four rotations visiting the port. Voyage edges are dashed and cargo transship directly from one voyage edge to another.

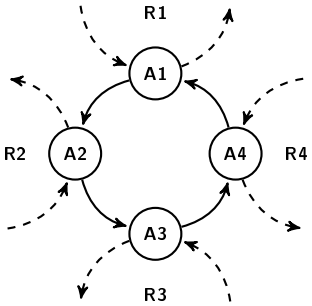

Figure 7: The transshipment structure shown for a physical port, A with four rotations visiting the port. $A1$, $A2$, $A3$, and $A4$ are the corresponding port calls and they are connected in a ring. Solid edges correspond to transshipment edges and dashed to voyage edges.

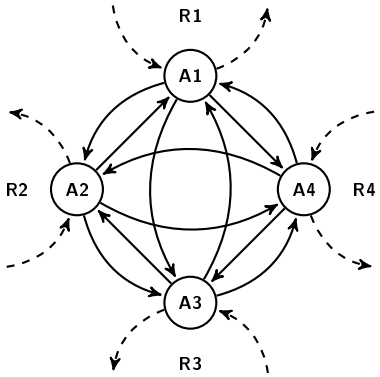

Figure 8: The *complete* transshipment structure shown for a physical port, A with four rotations visiting the port. $A1$, $A2$, $A3$, and $A4$ are the corresponding port calls. Solid edges correspond to transshipment edges and dashed to voyage edges for the four different rotations visiting port A.

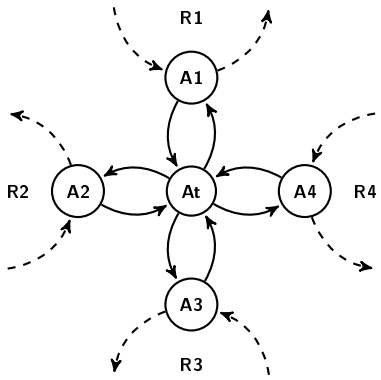

Figure 9: The *star* transshipment structure shown for a physical port, A with four rotations visiting the port. $A1$, $A2$, $A3$, and $A4$ are the corresponding port calls and At is an extra transshipment node. Solid edges correspond to transshipment edges and dashed to voyage edges for the four different rotations visiting port A.

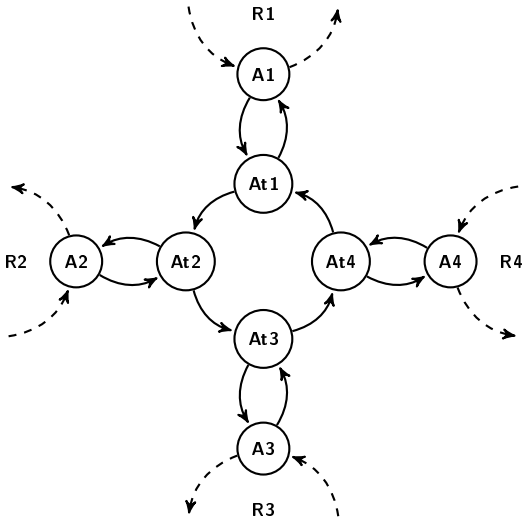

Figure 10: The *ring* transshipment structure shown for a physical port, A with four rotations visiting the port. $A1$, $A2$, $A3$, and $A4$ are the corresponding port calls. $At1$, $At2$, $At3$, and $At4$ are extra transshipment nodes. Solid edges correspond to transshipment edges and dashed to voyage edges for the four different rotations visiting port A.

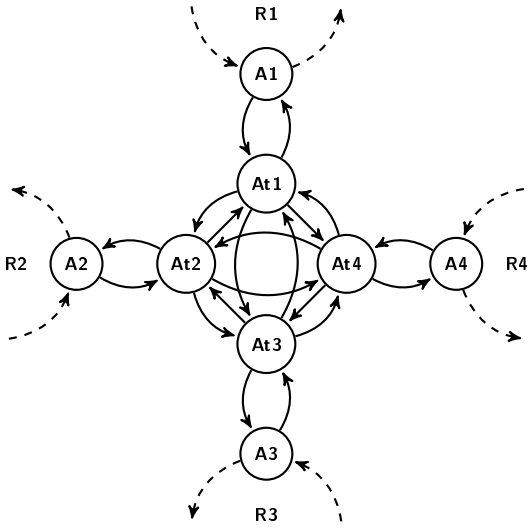

Figure 11: A general transshipment structure shown for a physical port, A, with four rotations visiting the port. $A1$, $A2$, $A3$, and $A4$ are the corresponding port calls. $At1$, $At2$, $At3$, and $At4$ are extra transshipment nodes and edges are added between all pairs of these. Solid edges correspond to transshipment edges and dashed to voyage edges.

6 Computational Experiments

The algorithms are implemented in `C++` and run on a normal laptop with an Intel Core i5 2.60GHz and 16 GB Ram using one core. We use the Boost Graph library to handle the networks and solve the LPs using the COIN-OR solver. We investigate the influence of the transit time limits, the graph contraction and different transshipment structures as well as sensitivity in the following. The results can be seen in Table 4 - 8 and Figure 12-15.

6.1 Data

The data instances used are based on the benchmark instances in *LINER-LIB 2012* (Brouer et al., 2013) published along with Brouer et al. (2014). We use networks constructed based on six of these instances, see Table 1. The networks have been constructed using the mat-heuristic that does not consider transit time described in Brouer and Desaulniers (2012). We report results for networks from each instance. These networks are denoted *Baltic, West Africa (WAF), Mediterranean (MED) Pacific, WorldSmall (WS0), AsiaEurope (AE0).* Furthermore, we consider additional large networks of varying quality, denoted *WorldSmall1 (WS1), WorldSmall2 (WS2), WorldSmall3 (WS3), AsiaEurope1 (AE1), AsiaEurope2 (AE2), AsiaEurope3 (AE3).* Table 2 shows the number of transshipment edges for the considered instances. For the instance AE0 the network consist of 308 voyage edges, 1530 transshipment edges, 616 load/unload edges, and 4000 forfeited edges and 111 ports and 308 rotation vertices corre-

Instance	Ports	Demands
Single hub instances		
Baltic	12	22
WAF	19	38
Multi hub instance		
MED	39	369
Trade lane instances		
Pacific	45	722
AE	111	4000
World instance		
WS	47	1764

Table 1: The instances considered. Consult Brouer et al. (2014) for further details.

	transshipment structure		
# transshipment edges	complete	star	ring
Baltic (13 voyage & 26 load edges)	22	12	17
WAF (43 voyage & 86 load edges)	166	64	98
MED (64 voyage & 128 load edges)	90	78	108
Pacific (153 voyage & 306 load edges)	734	278	410
WS0 (275 voyage & 550 load edges)	2076	534	797
AE0 (308 voyage & 616 load edges)	1530	526	773

Table 2: Number of transshipment edges for the different structures for the two largest instances. The first column gives the number of transshipment edges for the complete transshipment structure, the second column correspond to the star structure, and the third column to the ring structure.

sponding to port calls, i.e., a total of 2454 edges (6454 including the forfeited edges) and 422 nodes. WS0 correspondingly has 2901 edges (4665 including the forfeited edges) and 322 nodes. Edge costs are calculated as described in Brouer et al. (2014) using the data given in *LINER-LIB 2014*.

Table 3 shows the number of voyage edges in the graph for different instances. For different commodities we get slightly different graphs and hence the number of edges varies for the commodities by a few edges. The first column is the average number of contracted edges in the reduced graph for the o-d implementation of the RCSP algorithm. The second column states the average number of contracted voyage edges when using the o-all implementation of the RCSP algorithm, while the last column gives the number of voyage edges in the *full* graph used in Brouer and Desaulniers (2012).

As seen in Table 2 and 3, it has a significant effect to contract edges in smaller instances, where the networks are less complex and only few of the ports are visited by several rotations. However, for larger networks there are only very few edges that can be contracted because the majority of the ports serve several rotations. In all instances the number of load and unload edges is reduced as discussed earlier. If it was possible to identify ports where transshipments are not allowed or possible it would be possible to omit these nodes and reduce the graph further. In the next sections we consider the effects of time limits, the

# voyage edges	reduced (o-d)	reduced (o-all)	full
Baltic	7	7	26
WAF	36	37	43
MED	49	51	64
Pacific	146	148	153
WS0	271	274	275
AE0	280	287	308

Table 3: The average number of voyage edges in the reduced graph for an o-d and o-all representation compared to the number of voyage edges in the full graph. Dijkstra and o-all RCSP uses the o-all representation for the reduced graph, whereas the o-d RCSP uses the o-d representation for the reduced graph. The full graph is the same for all algorithms.

implementation of the shortest path algorithm, the transshipment structure, the graph reduction and finally the sensitivity of the time limits. For all instances we report the computational run times in seconds to solve the full MCF-problem to optimality, i.e., no more columns with reduced cost are found for any commodity.

6.1.1 Effect of Transit Time Limits and Implementation

Imposing realistic limits on the transit time for the individual commodities actually has a significant positive effect on the computational tractability. Even though it requires the solution of a more complex RCSP evaluation as subproblem, the vastly reduced solution space yields faster computations in almost all instances than when using Dijkstra's algorithm for the unconstrained problem. See Table 4 and Table 5 for a comparison of the instances. It is clear that the RSCP is only faster when implemented to take advantage of the problem structure.

Table 6 compares the implementation of RCSP as an *origin-destination* (o-d) implementation where the MCF problem is solved for all origin-destination pairs and an *origin-all* (o-all) implementation where all commodities with same origin is considered in one iteration of the RCSP-algorithm. In both cases we solve the problem to optimality considering all demands. Clearly the o-all implementation is advantageous with speed-ups in all larger instances and all discussed transshipment structures up to a factor of 9. The average speed-up for the considered instances using the full graph is 7 and 4 for the reduced graph. For the setting used in (Brouer and Desaulniers, 2012) with the complete transshipment structure and the full graph, the average speed-up is 2 when using the o-all RCSP compared to Dijkstra's algorithm, see Table 4 and Table 5 for a comparison of the instances. For the o-d implementation, the vast majority of the time is spent solving the sub-problem. For the o-all implementation for most instances more than half of the time is spent adjusting and solving the LP. Looking at Table 6 the o-all implementation with limits on transit time yields on average a speed-up of 1.5 for the considered larger instances on the reduced graph compared to the full graph.

Table 4 shows that the RSCP is only faster to solve when tight time limits are indeed imposed. The left part of the table shows run times and number of column generation iterations for instances where time limits are not imposed

RCSP o-all	No transit time limits					With transit time limits					RCSP Calls
	Time (s) star	It	Time (s) complete	It	Vol (%)	Time (s) star	It	Time (s) complete	It	Vol (%)	
	Full Graph										
Baltic	0.002 / 0.001	2	0.002 / 0.001	2	92.1	0.001 / 0.001	2	0.001 / 0.001	2	92.1	19
WAF	0.017 / 0.012	6	0.023 / 0.016	6	94.9	0.005 / 0.003	3	0.007 / 0.004	3	65	19
MED	0.213 / 0.095	6	0.214 / 0.099	6	95.3	0.065 / 0.037	3	0.066 / 0.039	3	60.5	36
Pacific	1.97 / 0.722	12	2.14 / 0.905	12	91.1	0.262 / 0.170	3	0.339 / 0.232	3	51.5	45
WS0	15.7 / 3.99	17	16.8 / 5.19	16	91.3	2.69 / 1.54	9	3.85 / 2.50	9	67.4	47
AE0	63.6 / 16.3	15	69.1 / 19.8	16	91.2	19.0 / 8.96	11	20.8 / 10.9	12	75.3	111
	Reduced Graph										
Baltic	0.001 / 0.001	2	0.002 / 0.001	2	92.1	0.001 / 0.001	2	0.001 / 0.001	2	92.1	19
WAF	0.012 / 0.005	6	0.017 / 0.009	6	94.9	0.005 / 0.002	3	0.006 / 0.003	3	65	19
MED	0.151 / 0.023	7	0.151 / 0.025	7	95.3	0.047 / 0.014	4	0.048 / 0.016	4	60.5	36
Pacific	1.44 / 0.203	13	1.58 / 0.343	12	91.1	0.154 / 0.059	3	0.227 / 0.114	3	51.5	45
WS0	13.0 / 0.99	17	14.1 / 2.16	16	91.3	1.79 / 0.56	9	2.82 / 1.34	9	67.4	47
AE0	48.2 / 1.82	15	47.9 / 3.14	17	91.2	12.0 / 1.80	11	13.2 / 2.92	11	75.3	111

Table 4: Comparison of run times (overall solution time / time spent in subproblem) when imposing limits on travel times for the full and reduced (contracted) graph with the star and complete transshipment structure. *It* indicates the number of column generation iterations to reach optimality. *Vol (%)* gives the volumes of cargo shipped. *RCSP Calls* gives the number of times the RCSP algorithm is called in each column generation iteration.

and the right part of the graph shows run times and number of column generation iterations with the limits imposed. On average for the considered instances of different size and structure the speed-up is 4 with the maximum being 9 when the time limits are imposed. Table 7 reveals that the share of containers shipped drops dramatically when transit times are imposed on networks designed without considering these. For several of the instances the utilization drops more than 30 percent point.

Dijkstra, full graph	Time (s) star	Time (s) complete	Vol (%)
Baltic	0.001	0.001	92.1
WAF	0.006	0.006	94.9
MED	0.115	0.102	95.3
Pacific	1.23	1.36	91.1
WS0	11.0	13.9	91.3
AE0	46.0	42.0	91.2

Table 5: Benchmark results for Dijkstra's algorithm, as in (Brouer and Desaulniers, 2012), for instances varying in size with no limits on travel time.

	Time (s) complete	Time (s) star	Time (s) ring	Time (s) complete	Time (s) star	Time (s) ring
	RCSP o-d			RCSP o-all		
	Full Graph					
AE1	132 / 118	104 / 90	128 / 109	20 / 8	18 / 6	24 / 7
AE2	100 / 90	82 / 74	100 / 87	13 / 6	13 / 5	16 / 6
AE3	118 / 108	95 / 83	116 / 103	16 / 7	14 / 6	19 / 7
WS1	21 / 20	13 / 12	16 / 14	3 / 2	2 / 1	3 / 1
WS2	26 / 25	13 / 13	15 / 14	3 / 2	2 / 1	2 / 1
WS3	29 / 28	16 / 15	20 / 17	3 / 2	2 / 1	3 / 1
	Reduced Graph					
AE1	63 / 50	37 / 23	44 / 26	15 / 3	13 / 2	18 / 2
AE2	46 / 38	27 / 18	37 / 25	10 / 2	9 / 1	12 / 2
AE3	48 / 39	27 / 18	39 / 24	11 / 2	10 / 1	14 / 2
WS1	10 / 9	4 / 3	6 / 4	2 / 1	1 / 0.3	2 / 0.4
WS2	13 / 11	4 / 3	7 / 5	2 / 1	1 / 0.4	2 / 1
WS3	16 / 14	6 / 4	8 / 6	2 / 1	2 / 0.4	2 / 1

Table 6: Comparison of the different transshipment structures, graph constructions and algorithms. The upper part of the table shows runtimes (overall solution time to reach optimality for the column generation procedure / time spent in sub-problem) for the full graph and the lower part shows results for the reduced (contracted) graph. Complete, star and ring refer to the different transshipment structures. The % volume shipped denotes the fraction of demand that can be satisfied when imposing limits on travel time. For comparison, the corresponding amounts without limits on travel time can be seen in Table 7.

6.1.2 Effect of Transshipment Structure

The effect of the different transshipment structures can be studied from Tables 6 and 4. Comparing the star transshipment structure to the complete structure reveals an average speed-up of less than 1.5. The ring structure only gives slight speed-up. However, the speed-up is more significant, when comparing the results for the reduced graph and the full graph adopted from (Brouer et al., 2014).

6.1.3 Effect of Reducing the Graph

The effect of contracting and reducing the number of edges in the graph can be observed from Tables 6 and 4. The main reduction comes from the reduced solution time of the sub-problems. The effect is clearly more pronounced for the larger instances, WS and AE, with an average overall speed-up above a factor of 2. The maximum speed-up is 3, while the average speed-up for the mid-size instances is only 1.3. The speed-up gained from reducing the graph is both a product of contracting the edges and removing the forfeited commodity edges included in (Brouer et al., 2014).

In general there are no unambiguous conclusion regarding the number of iterations in the column generation, but the column generation takes longer time for networks with a high percentage of volume shipped and for larger networks.

	Vol (%)	
	with transit time limits	no transit time limits
AE1	76.0	92.1
AE2	70.0	95.3
AE3	77.0	91.1
WS1	57.6	94.9
WS2	54.8	91.3
WS3	61.9	91.2

Table 7: The volumes shipped in the large instances when considering transit time limits compared to the volumes when transit time limits are not imposed.

Additionally, disregarding the forfeited commodity edges used in (Brouer and Desaulniers, 2012) in the RCSP implementation is important and for smaller instances the graph contraction makes the problem easier to solve. However, keeping the forfeited edges for the commodities in question aids bounding the algorithm.

6.1.4 Sensitivity of Travel Time Restrictions

Clearly the limits on travel time affect the size of the solution space and hence the amount of cargo that can flow through the network. Figure 12 shows how varying the limits between 80% and 200% of the limits given in *LINER-LIB 2014*, affects the amount of cargo with feasible paths in the 6 large instances described in Table 1. It is important to notice that the network optimization is done without considering travel times. At the default allowed travel time ($\alpha = 1.0$) as little as 51% of the goods can be transported in one instance, while up to around 90% of the goods can be transported when doubling the allowed travel times, see Table 8 or Figure 12.

In an optimized network of the type WS (WS0) where around 91 % of the goods can be transported when limits on travel times are not considered, the implications of time constraints is investigated further. Figure 13 shows that a slight increase of the volume of goods that can be transported can be obtained through the network when adjusting the time limits by a factor of 2.5 to 10. Figure 14 shows the revenue that can be obtained through the network, when adjusting the limits by a factor of 0.9 to 2.0. At the default allowed travel time ($\alpha = 1.0$) only around 60% of the maximum revenue (obtained without restrictions on travel time) can be obtained, while around 99% of the unconstrained revenue for this network can be obtained when doubling the allowed travel times. Figure 15 shows that a slight increase in revenue can be obtained, when adjusting the limits by a factor of 2.5 to 10. This clearly shows that some commodities will take very long undesirable paths through the network if there are no limits on travel time. A summary for the instances of different size can be seen in Table 8.

Note that the percentage of volume shipped as a function of allowed travel time is not necessarily a monotonically increasing function, whereas the revenue as a function of allowed travel time is. This is because more profitable cargo can become available on alternative paths as the time limits are increased. Furthermore, note that the effect is more significant for larger instances than smaller

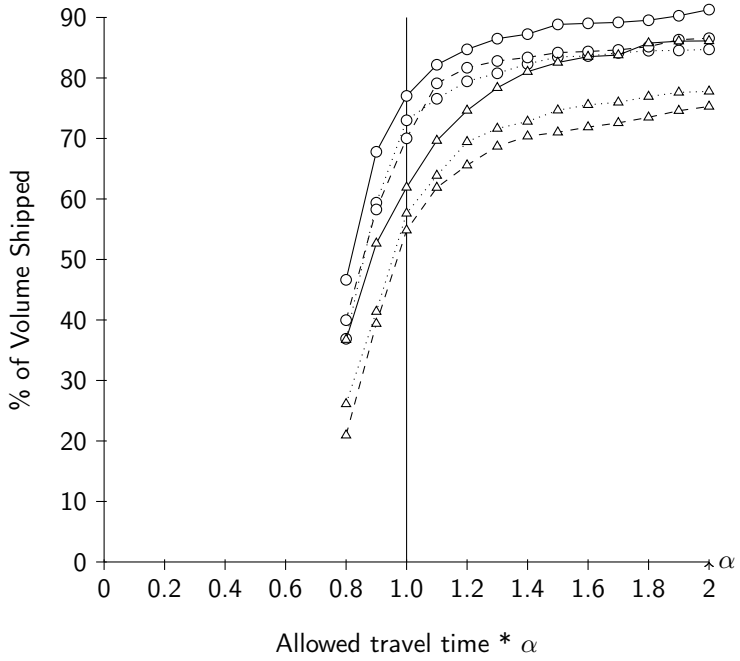

Figure 12: Sensitivity of limits on travel time based on the data given in (Brouer et al., 2014). The three upper instances (circles) are AE1 (dotted), AE2 (dashed), and AE3 (solid). The three lower instances (triangles) are WS1 (dotted), WS2 (dashed), and WS3 (solid).

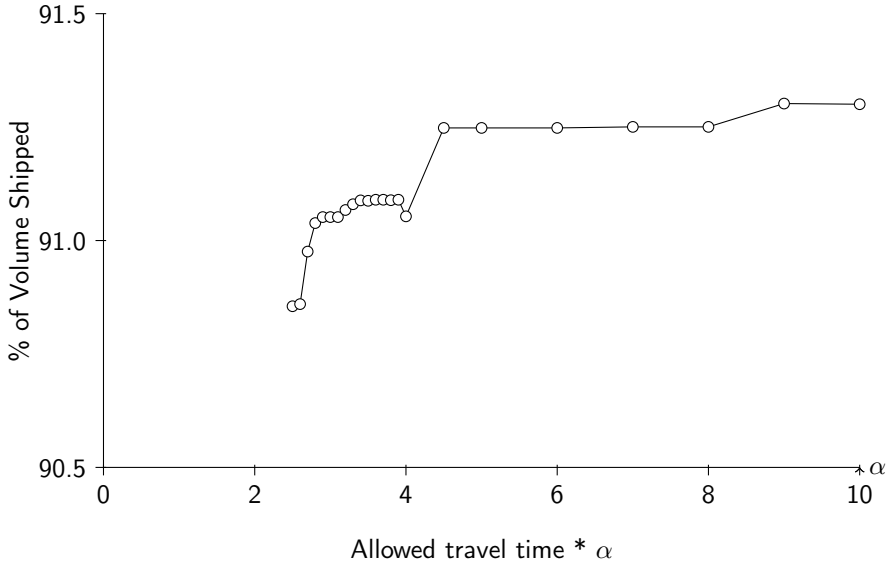

Figure 13: Sensitivity of limits on travel time based on the data for an instance of WS (WS0) given in (Brouer et al., 2014).

instances.

7 Conclusion

The presented analysis clearly shows that it is relevant and necessary to consider limits on travel times in the network design process. Omitting the transit time constraint when designing routes lead to cargo being transported along intricate routes that would not be accepted in practice. It could be feared that including transit time constraints in the MCF problem would lead to much higher computational times, but the present experiments show that this is not the case for the instances under study. The proposed graph contractions and simpler transshipment structures further help speeding up the solution time of the time constrained multi-commodity network flow problem. The obvious next step is to include the proposed algorithm for the time constrained multi-commodity network flow problem into heuristics for solving the liner shipping network design problem.

One should also take into account, that when designing a liner shipping network, the actual departure times (the schedule) are not fixed yet, meaning that transshipment times are only estimates. Hence, instead of using a very tight constraint on the transshipment time, a soft punishment could be used for exceeding the maximum allowed transshipment time up to a given upper limit. This could e.g. be a quadratic punishment also giving a reward for transshipment times below the limit. The punishment somehow indicates how difficult it will be to subsequently design a schedule that meets the time constraints. This is all easily handled by having a complete list of transit times and costs from solving the shortest path problem using a dynamic programming algorithm.

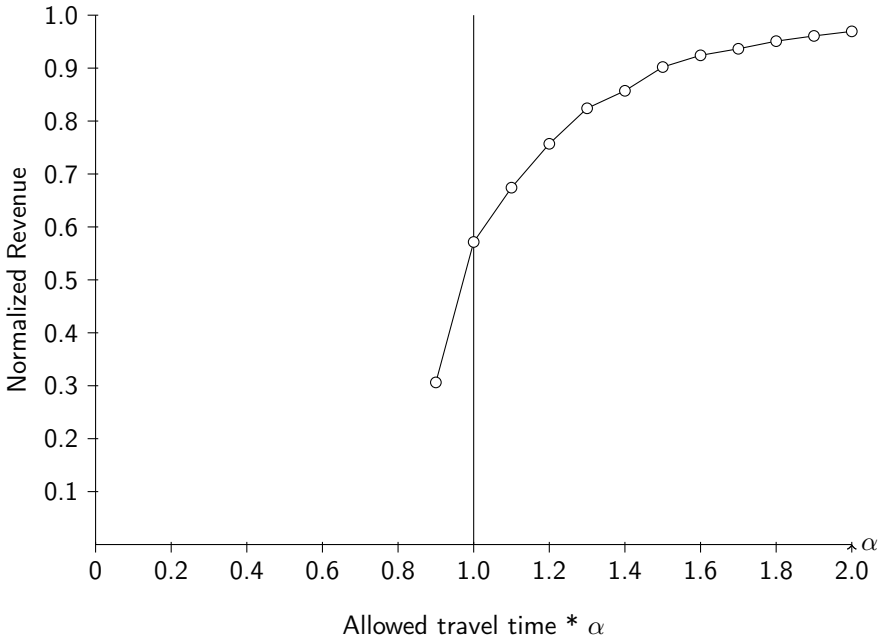

Figure 14: Sensitivity of limits on travel time based on the data for an instance of WS (WS0) given in (Brouer et al., 2014).

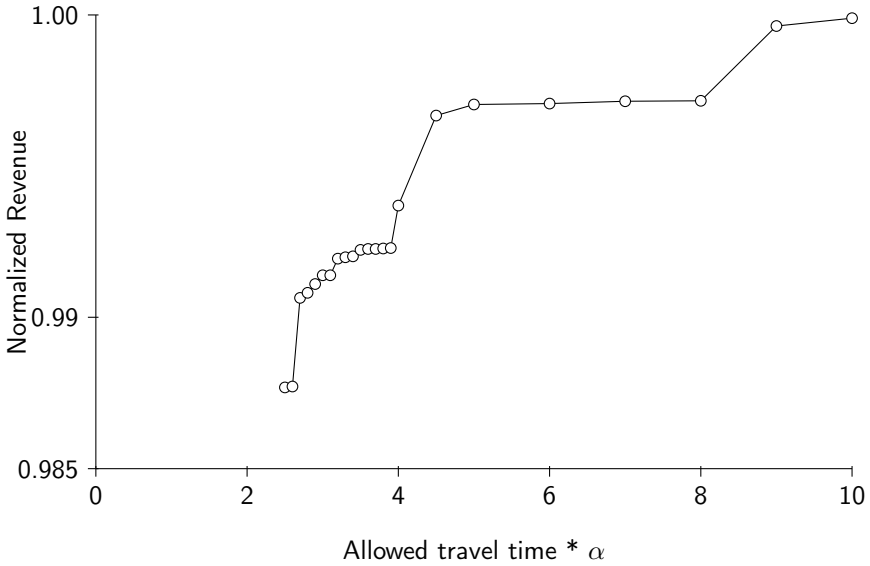

Figure 15: Sensitivity of limits on travel time based on the data for an instance of WS (WS0) given in (Brouer et al., 2014).

Instance	1 · time	2 · time	20 · time
Baltic (vol.)	1 (92%)	1 (92%)	1 (92%)
Baltic (rev.)	1	1	1
WAF (vol.)	0.67 (65%)	1.00 (95%)	1 (95%)
WAF (rev.)	0.42	0.99	1
MED (vol.)	0.63 (61%)	0.90 (86%)	1 (95%)
MED (rev.)	-0.76	0.51	1
Pacific (vol.)	0.57 (51%)	0.97 (88%)	1 (91%)
Pacific (rev.)	-0.30	0.92	1
WS (vol.)	0.74 (67%)	0.98 (89%)	1 (91%)
WS (rev.)	0.57	0.97	1
AE (vol.)	0.83 (75%)	0.96 (88%)	1 (91%)
AE (rev.)	0.76	0.96	1

Table 8: Normalized volumes transported (vol.) and normalized revenues (rev.) for the different instances under different transit time limits and compared to no transit time limits imposed. In column two, three and four, numbers in parenthesis indicate the absolute amount of goods transported. The first column shows the implication of the actual transit time limit, while we allow twice the time in column two and 20 times the allowed time in column three. 20 times the allowed time, in practice corresponds to no restrictions on travel time.

Acknowledgements

The authors wish to thank Christian Plum at Maersk Line and three anonymous referees for valuable comments. This project was supported in part by The Danish Strategical Research Council and The Danish Energy Technology Development and Demonstration Program (EUDP) under the ENERPLAN and GREENSHIP project and in part by The Danish Maritime Fund under the Competitive Liner Shipping Network Design project.

References

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

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice hall.

Alvarez, J. (2009). Joint routing and deployment of a fleet of container vessels. *Maritime Economics & Logistics*, 11:186–208.

Álvarez, J. F. (2011). Mathematical expressions for the transit time of merchandise through a liner shipping network. *Journal of the Operational Research Society*, 63(6):709–714.

Brouer, B., Alvarez, J., Plum, C., Pisinger, D., and Sigurd, M. (2013). Liner-lib 2012.

Brouer, B., Alvarez, J., Plum, C., Pisinger, D., and Sigurd, M. (2014). A base integer programming model and benchmark suite for liner shipping network design. *Transportation Science*, 48(2):281–312.

Brouer, B. and Desaulniers, G. (2012). A matheuristic for the liner shipping network design problem. In *Liner Service Network Design (PhD thesis)*, chapter 5. Technical University of Denmark, Department of Management Engineering.

Brouer, B., Pisinger, D., and Spoorendonk, S. (2011). The cargo allocation problem with empty repositioning (caper). *INFOR*, 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.

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

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2005). *Column generation*, volume 5. Springer.

Ducruet, C. (2013). Network diversity and maritime flows. *Journal of Transport Geography*, 30:77–88.

Ducruet, C. and Notteboom, T. (2012). The worldwide maritime network of container shipping: spatial structure and regional dynamics. *Global Networks*, 12(3):395–423.

Garey, M. R. and Johnson, D. S. (1979). Computers and intractability: a guide to the theory of np-completeness. *WH Freeman & Co., San Francisco*.

Gelareh, S., Nickel, S., and Pisinger, D. (2010). Liner shipping hub network design in a competitive environment. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):991–1004.

Hassin, R. (1992). Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42.

Holmberg, K. and Yuan, D. (2003). A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1):42–57.

IMO (2014). International maritime organization (imo).

Irnich, S. and Desaulniers, G. (2005). *Shortest path problems with resource constraints*. Springer.

Kaluza, P., Kölzsch, A., Gastner, M. T., and Blasius, B. (2010). The complex network of global cargo ship movements. *Journal of the Royal Society Interface*, 7(48):1093–1103.

Maersk (2014). The maersk line service network.

MaritimeCO2 (2014). Impact assessment for the adoption of co2 emission trading for maritime transport.

Meng, Q. and Wang, S. (2012). Liner ship fleet deployment with week-dependent container shipment demand. *European Journal of Operational Research*, 222(2):241–252.

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.

Notteboom, T. E. (2006). The time factor in liner shipping services. *Maritime Economics & Logistics*, 8(1):19–39.

Notteboom, T. E. and Vernimmen, B. (2009). The effect of high fuel costs on liner service configuration in container shipping. *Journal of Transport Geography*, 17(5):325–337.

Plum, C. E., Pisinger, D., Salazar-González, J.-J., and Sigurd, M. M. (2014). Single liner shipping service design. *Computers & Operations Research*, 45:1–6.

Plum, C. E., Pisinger, D., and Sigurd, M. M. (2013). A service flow model for the liner shipping network design problem. *European Journal of Operational Research*.

Wang, S. and Meng, Q. (2011). Schedule design and container routing in liner shipping. *Transportation Research Record: Journal of the Transportation Research Board*, 2222(1):25–33.

Wang, S. and Meng, Q. (2012). Liner ship route schedule design with sea contingency time and port time uncertainty. *Transportation Research Part B: Methodological*, 46(5):615–633.

Wang, S. and Meng, Q. (2013). Reversing port rotation directions in a container liner shipping network. *Transportation Research Part B: Methodological*, 50:61–73.

Wang, S. and Meng, Q. (2014). Liner shipping network design with deadlines. *Computers & Operations Research*, 41:140–149.

Wang, S., Meng, Q., and Sun, Z. (2013). Container routing in liner shipping. *Transportation Research Part E: Logistics and Transportation Review*, 49(1):1–7.