Routing and Scheduling in Tramp Shipping - Integrating Bunker Optimization

Technical report

Vilhelmsen, Charlotte; Lusby, Richard Martin ; Larsen, Jesper

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

Routing and Scheduling in Tramp Shipping - Integrating Bunker Optimization

Charlotte Vilhelmsen
Department of Management Engineering, Technical University of Denmark
Produktionstorvet, Building 426, 2800 Kgs. Lyngby, Denmark
chaan@dtu.dk

Richard Lusby
Department of Management Engineering, Technical University of Denmark
Produktionstorvet, Building 426, 2800 Kgs. Lyngby, Denmark
rmlu@dtu.dk

Jesper Larsen
Department of Management Engineering, Technical University of Denmark
Produktionstorvet, Building 426, 2800 Kgs. Lyngby, Denmark
jesla@dtu.dk

February 11, 2013

Abstract

A tramp ship operator typically has some contracted cargoes that must be carried and seeks to maximize profit by carrying optional cargoes. Hence, tramp ships operate much like taxies following available cargoes and not according to a fixed route network and itinerary as liner ships. Marine fuel is referred to as *bunker* and bunker costs constitute a significant part of the daily operating costs. There can be great variations in bunker prices across bunker ports so it is important to carefully plan bunkering for each ship. As ships operate 24 hours a day, they must refuel during operations. Therefore, route and schedule decisions affect the options for bunkering. Current practice is, however, to separate the two planning problems by first constructing fleet schedules and then plan bunkering for these fixed schedules. In this paper we explore the effects of integrating bunker planning in the routing and scheduling phase and present a mixed integer programming formulation for the integrated problem of optimally routing, scheduling and bunkering a tramp fleet. Aside from the integration of bunker, this model also extends standard tramp formulations by using load dependent costs, speed and bunker consumption. We devise a solution method based on column generation with a dynamic programming algorithm to generate columns. The method is heuristic mainly due to a discretization of the continuous bunker purchase variables. We show that the integrated planning approach can increase profits and that the decision of which cargoes to carry and on which ships is affected by the bunker integration and by changes in the bunker prices.

Keywords: Tramp Shipping, Routing, Scheduling, Bunkering, Column Generation.

1 Introduction

It is estimated that over 80% of world trade is carried by the international shipping industry (UNCTAD [2011]) and world trade therefore depends on the industry's efficiency and competitive freight rates. Hence, research to increase efficiency within maritime transportation is important, and, taking the mere size of this huge industry into consideration, even small improvements can have great impact.

An important part of utilizing the existing fleet efficiently is routing and scheduling the ships, i.e. assigning cargoes to ships while simultaneously finding the sequence and timing of port calls

for all ships. Many ship operators use experienced planners to manually route and schedule the fleet. However, increased competition and recent trends of mergers among and pooling of shipping companies have increased the pressure as well as the difficulty of devising efficient schedules manually due to the increased fleet sizes (Christiansen et al. [2004]). Therefore, there is a need for an automated approach to the planning that can both aid the construction of efficient schedules and enable fast changes to existing schedules in case of new opportunities or changed demand.

In this paper we focus on tramp shipping where ships operate much like taxies following the available cargoes and not according to a fixed route network and itinerary as in liner shipping. Routing and scheduling within tramp shipping is therefore a more dynamic and ongoing process compared to that of liner shipping. A tramp operator often has some contracted cargoes that must be carried and seeks to maximize profit by carrying optional cargoes found in the spot market.

Despite the above motivation, routing and scheduling within maritime transportation has not received as much attention as similar land based transportation problems (Christiansen et al. [2004]). Research in this area has, however, increased in recent years. Tramp ship routing and scheduling is very closely related to the well researched Vehicle Routing Problem (VRP) and its variants but there are, however, important differences that facilitate the development of tramp ship specific methods. To mention a few, we note that optional cargoes are not considered in the standard VRP and that ships pay port fees and operate around the clock.

Marine fuel is also referred to as *bunker fuel* or simply *bunker* while refueling is called *bunkering*. Fuel costs constitute a significant part of daily operating costs and since bunker prices can vary significantly across ports, it is important to carefully plan the bunkering of each ship. The recent increase in oil prices adds further motivation for operators to plan bunkering optimally, yet many still use manual planning. Ships operate 24 hours a day so they must refuel during operations. Hence, route and schedule decisions will affect the options for bunkering. Consequently, it seems natural to integrate bunker planning in the routing and scheduling phase and consider the combined routing, scheduling and bunkering problem. Current practice is, however, to separate the two problems by first constructing fleet schedules and then plan bunkering for these fixed schedules.

In this paper we explore the effects of integrating bunker planning in the routing and scheduling phase. We present a mixed integer programming formulation for the integrated problem of optimally routing, scheduling and bunkering a tramp fleet. This model extends standard tramp formulations by accounting for bunkering time, variations in bunker prices and bunker ports costs and further by using load dependent costs as well as speed and bunker consumption. We devise a heuristic solution method that can simultaneously select which optional cargoes to carry, how cargoes should be allocated to ships, determine ship routes and schedules, and decide when, where and how much each ship should bunker during its schedule depending on forward curves for bunker prices. The method relies on column generation with a dynamic programming algorithm to generate columns. Computational results show that this integrated planning approach can increase profits for tramp operators, and that the decision of which cargoes to carry and on which ships is affected by the bunker integration and by changes in the bunker prices.

The remainder of the paper is organized as follows. In Section 2 relevant literature is presented. Section 3 provides a problem description as well as a mathematical model for the problem, while the devised solution method is described in Section 4. Section 5 describes some instance generators that we have developed to acquire necessary data on cargoes and bunker prices. In Section 6 we tune the devised algorithm and in Section 7 we explore the effects and benefits of integrating bunker planning in the routing and scheduling phase through a comparison of the integrated approach and the sequential approach. We also investigate the method's sensitivity to bunker prices. Finally, concluding remarks and suggestions for future work are discussed in Section 8.

2 Literature review

Mathematical formulations and discussions on solution methods for a wide range of maritime problems on all planning levels can be found in Christiansen et al. [2007]. Furthermore, a thorough review of literature focused on ship routing and scheduling before 2005 can be found in the three review papers, Ronen [1983], Ronen [1993], and Christiansen et al. [2004].

Recent work on tramp ship routing and scheduling include Fagerholt and Lindstad [2007] who

describe a decision support system based on a multi-start local search heuristic. Korsvik et al. [2010] use tabu search to solve the problem and show significant improvements on large and tightly constrained cases compared to the multi-start local search heuristic. Malliappi et al. [2011] solve the problem using a variable neighborhood search. Several extensions of the standard tramp ship routing and scheduling have also been considered. E.g. Brønmo et al. [2007], Brønmo et al. [2010] and Korsvik and Fagerholt [2010] all consider flexible cargo sizes while Fagerholt et al. [2010], Norstad et al. [2011] and Gatica and Miranda [2011] all consider speed as a decision variable. Kobayashi and Kubo [2010] and Lin and Liu [2011] consider simultaneous cargo allocation for a problem with multiple and non compatible cargo products. Finally, project shipping as a segment of tramp shipping is explored in Andersson et al. [2011b] and Fagerholt et al. [2011] while split loads are considered in Korsvik et al. [2011], Andersson et al. [2011a] and Stålhane et al. [2012].

The tramp ship routing and scheduling problem is closely related to vehicle routing problems. Most similar to our problem is the vehicle routing problem with pickup and deliveries and time windows (VRPPDTW). We refer the reader to Desaulniers et al. [2002] for a problem description and a discussion on solution methods. There are, however, important differences between the maritime version of the problem and the land based one, creating the need for tailor made models and solution methods for the maritime industry. Ronen [1983], Ronen [2002] and Christiansen et al. [2004] elaborate on these differences but to mention a few, we note that ships pay port fees and operate continuously. Hence, the ships have different starting positions and starting times, as some ships can be occupied with prior tasks at the beginning of the planning period. Even in the multi-depot version of the VRPPDTW vehicles must return to their home depot whereas ships do not have to return to their starting point. Finally, the distinction between contract cargoes and optional cargoes leads to a priority on cargoes that is not used in VRPPDTW where all customers must be serviced at minimum cost. In contrast, the tramp objective is to maximize profit as in the less known Pickup and Deliver Selection Problem, see Schönberger et al. [2003].

A solution method that has received much attention and achieved great success within vehicle routing is column generation. This method is not as frequently used within maritime transportation. This is partly because the large number of constraints reduce the solution space to such an extent that - combined with the major uncertainty within maritime transportation - the possible schedules for each ship only consists of a few voyages. Hence, all feasible combinations can be enumerated and, so, it is often sufficient to apply a priori column generation. However, within tramp ship scheduling both Appelgren [1969] and Brønmo et al. [2010] have found dynamic column generation more efficient than a priori column generation. Brønmo et al. [2010] explore flexible cargo sizes and use branch-and-price to solve the problem. They discretize the cargo quantities to obtain a subproblem that is a shortest path problem with pickup and delivery and time windows, and use dynamic programming to solve it. They report that in their experiments, the dynamic approach is both faster and enables them to deal with larger or more loosely constrained instances than the a priori generation approach. In line with that, recent years have shown an increase in the number of maritime papers that explore dynamic column generation, see e.g. Kobayashi and Kubo [2010], Hennig et al. [2012] and Stålhane et al. [2012]. Furthermore, Desauliniers et al. [2005] devoted two whole chapters to column generation in maritime problems.

Within vehicle routing examples of theoretical research on refueling policies can be found in Hong Lin et al. [2007] and Lin [2008]. Aside from the theoretical work, software products for refueling, called *fuel optimizers*, have been developed for the trucking industry. In Suzuki [2008] a description of these systems as well as a literature review is given while Suzuki and Dai [2012] discuss solution methods. These systems use the latest price data to calculate which truck stops to use and how much fuel to purchase at each stop to minimize refueling costs. All the above work on refueling policies are for single vehicles traveling on a fixed route. Hence, there is no integration of refueling into the routing and scheduling phase. Also, since customers have already been assigned to vehicles, the interdependency of vehicle routes is ignored and the problem is decomposed into independent one-vehicle problems while we have to consider the entire fleet at once.

Within air transportation work on refueling policies can be found in Darnell and Loflin [1977], Stroup and Wollmer [1992], Abdelghany et al. [2005] and Zouein et al. [2002]. However, they also consider refueling policies for fixed routes and do not allow aircrafts to divert from their routes for refueling. In fact, since routes are fixed, the refueling policy problem relates more to liner shipping where the combinatorial aspect, i.e. the route selection, from tramp shipping is not present.

Within liner shipping Yao et al. [2012] explore refueling policies that incorporate sailing speed as a decision variable. As they consider a liner service, they too assume a fixed route and do not allow ships to divert from their routes for refueling. Similarly, Besbes and Savin [2009] also explore refueling policies for liner ships for a fixed route. In contrast, Notteboom and Vernimmen [2009] consider the impact of increasing bunker prices on the actual design of liner services.

Within tramp shipping, as far as we know, only two papers address the issue of optimal refueling. Oh and Karimi [2010] consider a multi parcel chemical tanker and propose a mixed integer programming model for finding an optimal bunker strategy. They incorporate speed as a decision variable and take uncertainty of fuel prices into account but again they assume a prefixed route. In contrast, Besbes and Savin [2009] instead find a profit maximizing refueling strategy while simultaneously optimizing routes. They formulate the problem as a stochastic dynamic program. Their setup is, however, completely different from the one considered in this paper as they approach the problem from a much more strategic level. They only consider one ship and do not consider any actual cargoes. Instead they assume a stochastic revenue process that leads them to explore optimal *cycles* in the network. In fact, they view the problem more as an inventory management problem with no end, and, consequently, seek to maximize the *long term average* profit. They characterize the optimal refueling policy when prices are constant over time and do not differ across ports and when prices are constant over time but differ across ports. However, they do not consider the case where prices vary over time and at the same differ across ports as we intend to do.

3 Problem Description

In this section we give a problem description starting with the pure tramp ship routing and scheduling problem. We then move on to include bunkering and present a mathematical model for the Tramp Ship Routing And Scheduling Problem with Bunker Optimization (TSRSPBO).

3.1 The Pure Tramp Ship Routing and Scheduling Problem

A tramp ship operator has some long term contracts that obligates him to carry some cargoes and can choose to carry additional cargoes, so called *spot cargoes*, if fleet capacity allows it and it is found to be profitable. The objective is to create a profit maximizing set of fleet schedules, one for each ship, where a schedule is a sequence and timing of port calls representing cargo loading and discharging. The optimal solution therefore combines interdependent, and hence simultaneous, decisions on which optional cargoes to carry, the assignment of cargoes to ships and the optimal sequence and timing of port calls for each ship.

A cargo is mainly characterized by the quantity to be transported, the revenue obtained from transporting it and the pickup and discharge port. There is also a service time for loading and discharging and a time window giving the earliest and latest start for loading. In some cases there is also a time window for discharge.

A tramp fleet is usually heterogeneous, comprised of ships of different sizes, load capacities, bunker consumptions, speeds, and other characteristics. Ships can be occupied with prior tasks when planning starts so each ship is further characterized by the time it is available for service and the location it is at when it becomes available. The characteristics of a ship determine which cargoes, ports and canals it is compatible with. E.g. the draft of a ship can prohibit it from entering a shallow port.

As we consider a fixed fleet we can disregard fixed costs. The main sailing cost is fuel cost and this is different for each ship and load dependent. In traditional tramp ship routing and scheduling models, sailing cost, and in turn bunker consumption, is assumed independent of the load of the ship. We, however, will not make such an assumption. When loading and discharging, ship dependent port costs are incurred. While loading and unloading, ships also consume bunker although much less than at sea. Other costs can be relevant depending on the specific operator.

A mathematical arc flow formulation for the pure routing and scheduling problem can be found in Christiansen et al. [2007]. It is formulated as a pickup and delivery problem with time windows and capacity constraints. The model, just as most other tramp ship routing and scheduling models, assumes that sailing costs are fixed with no consideration to the great variations in bunker prices

or the port costs incurred when bunkering. Furthermore, the time consumption of bunkering is not considered. In our work, we integrate considerations for bunker price variations, bunker port costs as well as the time aspect of bunkering and extend the pure routing and scheduling formulation to include variables for bunker purchases for each ship, new constraints to incorporate these variables and, finally, an extended objective function that reflects this new way of calculating bunker costs, and in turn sailing costs. We also incorporate load dependent bunker consumption. Bunker consumption also depends on the speed of the ship. However, in reality the speed is not necessarily fixed but instead allowed to vary with the load of the ship. E.g. if a ship sails at 'full speed', the actual speed depends on the load of the ship. Likewise, a speed setting often incurred is 'ECO speed', i.e. the most economical speed, and this speed also depends on the load of the ship. We assume each ship sails at 'ECO speed' and, so, the actual speed and bunker consumption depend on the load of the ship. Load dependent bunker consumption means that costs are also load dependent which further allows us to use load dependent port and canal costs.

3.2 Incorporating Bunker

Bunkering can take place at a bunker port where ships enter port just to refuel or at a pickup or discharge port since almost all ports involved in shipping also sell bunker. Bunkering at a pickup or discharge port has the obvious benefit of avoiding detours just for bunker, only incurring port costs once and even saving time if concurrent bunkering is allowed. However, price variations between ports can easily be large enough to compensate for the extra cost of a detour, the extra port costs, and also the extra time consumption. In fact, a few ports seem to dominate bunker sales because of their strategic location along major trade routes thereby limiting the detours necessary for ships to refuel there. Two examples of such ports are Malta and Singapore (Oh and Karimi [2010]).

A bunker option is mainly characterized by its geographical location, port costs, bunker price and time window in which this price is assumed to remain valid. Several bunker options can represent the same physical bunker port but at different times and, hence, with different prices. Time dependent port costs or port opening hours can also cause a separation of one bunker port into many bunker options with different prices and associated time windows. Due to high volatility in bunker prices these time windows are bound to be narrow and without loss of generality we assume they are so narrow that no ship will use the same bunker option twice. If time windows are not narrow enough for such an assumption, each time window can simply be split into several smaller time windows each with an associated bunker option until the assumption is valid.

Ships operate 24 hours a day so there is no natural end to the optimization problem. Hence, the condition of the fleet at the end of the planning period affects optimization in the next period. With bunker included in the optimization process, the initial bunker inventory for each ship is therefore an important part of fleet data. Likewise, any remaining bunker onboard ships at the end of their schedules must be considered a valuable resource for the next planning period independent of future demand. To account for this resource, we put a premium on any quantity above the initial bunker level for each ship and call this quantity *bonus bunker*. Similarly, ships that end their schedule with less bunker than the initial level must pay for using this resource. As we discuss later, it is in fact a vital part of the solution method that this bonus bunker is accounted for. However, the actual value of it is difficult to price. Using the price of the last visited bunker port is not possible for ships that due to a high initial bunker inventory or idleness did not refuel during their schedule. It also leads to arbitrage since visiting an expensive bunker port last to purchase a small amount will drive up the resell price even though the bonus bunker might have been bought at a really cheap bunker port. Likewise, there is no incentive to fill up the tank if a ship passes a cheap bunker port last. Allowing bonus bunker to be resold at an average price of the region that the ship ends its schedule in motivates repositioning ships to regions with high bunker prices with no consideration for future cargo demand. Therefore, we calculate premiums for bonus bunker at an average price of all bunker options with time windows containing the end of the planning horizon, i.e. a geographically independent forecast of the average bunker price at the end of the planning horizon. Ship data will now also include a minimum and maximum bunker level corresponding to a required safety level and tank capacity, respectively.

Concurrent bunkering can easily be added to the model but we have chosen not to include it and further chosen to assume that all bunker options have the same pumping rate for bunkering.

The reason is that we want to be able to differentiate bunker options on their prices and geographic location rather than their 'timing' as this allows us to explore the solution's sensitivity to changes in bunker prices. Furthermore, when taking the time for port clearance, berthing etc. into account, the time spent actually pumping bunker onboard the ships can be considered negligible and considering the trade off between solution time and complexity we have chosen to assume a fixed time for bunkering at all options regardless of the amount purchased.

3.3 Mathematical model

Let $\mathcal{V}$ be the set of ships in the fleet and index it by v, and let $\mathcal{B}$ denote the set of bunker options indexed by k. Since not all ships are compatible with all ports, we get ship specific bunker sets denoted $\mathcal{B}^v \subseteq \mathcal{B}$. Furthermore, we assume that there are N cargoes and index them by i.

Let $\mathcal{N}_P = \{1, \cdots, N\}$ and $\mathcal{N}_D = \{N+1, \cdots, 2N\}$ denote the set of pickup and discharge nodes respectively. We represent each cargo i by a pickup node $i \in \mathcal{N}_P$ and a discharge node $N+i \in \mathcal{N}_D$. We define $\mathcal{N} = \mathcal{N}_P \cup \mathcal{N}_D$ as the set of all cargo related nodes and partition $\mathcal{N}_P$ into $\mathcal{N}_P = \mathcal{N}_C \cup \mathcal{N}_O$, where $\mathcal{N}_C$ and $\mathcal{N}_O$ contain pickup nodes for contract cargoes and optional cargoes, respectively. Associated with each ship v is now a standard network $(\mathcal{N}^v, \mathcal{A}^v)$ not including bunker options. The standard network nodes, $\mathcal{N}^v \subseteq \mathcal{N}_P \cup \mathcal{N}_D \cup \{o(v), d(v)\}$, correspond to cargoes that ship v is able to carry and two ship specific nodes representing, respectively, the origin and an artificial destination for ship v. Ship v is able to carry a cargo i if it has sufficient capacity, is compatible with the specific load and discharge ports and is in general compatible with cargo i on all accounts. The ship specific cargo nodes are given by $\mathcal{N}_P^v = \mathcal{N}_P \cap \mathcal{N}^v$ for pickups and $\mathcal{N}_D^v = \mathcal{N}_D \cap \mathcal{N}^v$ for discharges. The set of standard network arcs, $\mathcal{A}^v$, is a subset of $\{(i,j) | i \in \mathcal{N}^v, j \in \mathcal{N}^v\}$ and contains all the arcs traversable by ship v, e.g. with respect to time and bunker consumption.

For each ship v we extend the standard node set, $\mathcal{N}^v$, by adding a node for each element in $\mathcal{B}^v$ and index the full set by i. Likewise, we extend the standard arc set, $\mathcal{A}^v$, by adding all arcs connecting nodes in $\mathcal{N}^v \setminus d(v)$ with nodes in $\mathcal{B}^v$ and traversable by ship v with respect to time and bunker. We do not connect the destination node, $d(v)$, with bunker nodes, as we do not want idle ships to bunker since this could send them in the exact opposite direction of their next (unplanned) port stop and since their unplanned voyages could involve port stops with very attractive prices. For each ship v we thereby obtain an extended cargo-bunker network $(\hat{\mathcal{N}^v}, \hat{\mathcal{A}}^v) = (\mathcal{N}^v \cup \mathcal{B}^v, \mathcal{A}^v \cup \mathcal{A}_B^v)$ where $\mathcal{A}_B^v$ denotes arcs connecting bunker nodes to nodes in $\mathcal{N}^v \setminus d(v)$.

For $v \in \mathcal{V}$ and $i \in \hat{\mathcal{N}^v}$ we define a continuous variable l_i^v that denotes the load onboard ship v just after completing service at node i. With $(i,j) \in \hat{\mathcal{A}}^v$ we then associate a load dependent time consumption $T_{ij}^v(l_i^v)$ when traversed by ship v carrying a load of l_i^v and calculated from the arrival at the port of node i until the arrival at the port of node j. It accounts for service time at the port of node i whether it is a loading, discharging or bunkering node, and the sailing time from the port of node i to the port of node j. We also associate a load dependent bunker consumption function $B_{ij}^v(l_i^v)$ that accounts for bunker consumption while traveling from node i to node j but not including bunker consumption while in port at node i. This port consumption is instead accounted for by B_i^v. Finally, we have the variable cost function $C_{ij}^v(l_i^v)$. Like time consumption, this accounts for costs related to visiting the port of node i and sailing costs from the port of node i to the port of node j. Note however, that the cost of purchasing bunker is not included, as it is a dynamic node cost dependent on the amount of bunker purchased at the node. Instead, this cost will be added separately and accounted at a bunker unit price of P_k for $k \in \mathcal{B}$ while bonus bunker is 'resold' at a unit price P. Also note that if nodes i and j correspond to the same physical port, $C_{ij}^v(l_i^v)$ does not include port costs, $T_{ij}^v(l_i^v)$ does not include travel time and $B_{ij}^v(l_i^v) = 0$. We also have a revenue R_i and a quantity Q_i for all cargoes $i \in \mathcal{N}_P$. We denote the cargo capacity of ship v by V_{CAP}^v and the bunker capacity by B_{Max}^v. The safety level for bunker inventory is denoted B_{Min}^v while the initial bunker level onboard the ship is denoted B_0^v. Finally, we denote by $[T_{MNi}^v, T_{MXi}^v]$ the time window associated with node $i \in \hat{\mathcal{N}^v}$. For $o(v)$ this window is collapsed into the time ship v is available for service. For the mathematical formulation we need the following variables:

x_{ij}^v, $v \in \mathcal{V}$, $(i,j) \in \hat{\mathcal{A}}^v$. Binary variable that is equal to 1, if ship v visits node i just before node j, and 0 otherwise

t_i^v, $v \in \mathcal{V}$, $i \in \hat{\mathcal{N}^v}$. Denotes the time ship v begins service at node i

$l^v_{Ci},\ v \in \mathcal{V}$. Denotes the cargo load onboard ship v just after completing service at node i

$l^v_{Bi},\ v \in \mathcal{V}$. Denotes the bunker load onboard ship v just after completing service at node i

$l^v_i,\ v \in \mathcal{V}$. Denotes the total load onboard ship v just after completing service at node i

$y^v_k,\ v \in \mathcal{V},\ k \in \mathcal{B}^v$. Gives the bunker purchased by ship v at option $k \in \mathcal{B}^v$

We can now give an arc flow formulation of the TSRSPBO:

$$\max \sum_{v\in\mathcal{V}} \sum_{i\in\mathcal{N}^v_P} R_i\Big(\sum_{j\in\hat{\mathcal{N}}^v} x^v_{ij}\Big) - \sum_{v\in\mathcal{V}} \sum_{(i,j)\in\hat{\mathcal{A}}^v} C^v_{ij}(l^v_i)x^v_{ij} - \sum_{v\in\mathcal{V}} \sum_{k\in\mathcal{B}^v} y^v_k P_k + \sum_{v\in\mathcal{V}} P\cdot(l^v_{Bd(v)} - l^v_{Bo(v)}) \tag{1}$$

s.t.

$$\sum_{v\in\mathcal{V}} \sum_{j\in\hat{\mathcal{N}}^v} x^v_{ij} = 1, \qquad \forall i \in \mathcal{N}_C, \tag{2}$$

$$\sum_{v\in\mathcal{V}} \sum_{j\in\hat{\mathcal{N}}^v} x^v_{ij} \le 1, \qquad \forall i \in \mathcal{N}_O, \tag{3}$$

$$\sum_{j\in\mathcal{N}^v_P\cup\mathcal{B}^v\cup\{d(v)\}} x^v_{o(v)j} = 1, \qquad \forall v \in \mathcal{V}, \tag{4}$$

$$\sum_{i\in\hat{\mathcal{N}}^v} x^v_{ij} - \sum_{i\in\hat{\mathcal{N}}^v} x^v_{ji} = 0, \qquad \forall v \in \mathcal{V},\ j \in \hat{\mathcal{N}}^v \setminus \{o(v), d(v)\}, \tag{5}$$

$$\sum_{i\in\mathcal{N}^v_D\cup\mathcal{B}^v\cup\{o(v)\}} x^v_{id(v)} = 1, \qquad \forall v \in \mathcal{V}, \tag{6}$$

$$x^v_{ij}(t^v_i + T^v_{ij}(l^v_i) - t^v_j) \le 0, \qquad \forall v \in \mathcal{V},\ (i,j) \in \hat{\mathcal{A}}^v, \tag{7}$$

$$T^v_{MNi} \le t^v_i \le T^v_{MXi}, \qquad \forall v \in \mathcal{V}, i \in \hat{\mathcal{N}}^v, \tag{8}$$

$$x^v_{ij}(l^v_{Ci} + Q_j - l^v_{Cj}) = 0 \qquad \forall v \in \mathcal{V}, (i,j) \in \hat{\mathcal{A}}^v | j \in \mathcal{N}^v_P, \tag{9}$$

$$x^v_{i,N+j}(l^v_{Ci} - Q_j - l^v_{C,N+j}) = 0 \qquad \forall v \in \mathcal{V}, (i,N+j) \in \hat{\mathcal{A}}^v | j \in \mathcal{N}^v_P, \tag{10}$$

$$x^v_{ij}(l^v_{Ci} - l^v_{Cj}) = 0 \qquad \forall v \in \mathcal{V}, (i,j) \in \hat{\mathcal{A}}^v | j \in \mathcal{B}^v, \tag{11}$$

$$l^v_{Co(v)} = 0 \qquad \forall v \in \mathcal{V}, \tag{12}$$

$$\sum_{j\in\hat{\mathcal{N}}^v} Q_i x^v_{ij} \le l^v_{Ci} \le \sum_{j\in\hat{\mathcal{N}}^v} V^v_{CAP} x^v_{ij} \qquad \forall v \in \mathcal{V},\ i \in \mathcal{N}^v_P, \tag{13}$$

$$0 \le l^v_{C,N+i} \le \sum_{j\in\hat{\mathcal{N}}^v} (V^v_{CAP} - Q_i) x^v_{N+i,j} \qquad \forall v \in \mathcal{V},\ i \in \mathcal{N}^v_P, \tag{14}$$

$$x^v_{ij}(l^v_{Bi} - B^v_{ij}(l^v_i) - B^v_j + y^v_j - l^v_{Bj}) = 0 \qquad \forall v \in \mathcal{V}, (i,j) \in \hat{\mathcal{A}}^v | j \in \mathcal{B}^v, \tag{15}$$

$$x^v_{ij}(l^v_{Bi} - B^v_{ij}(l^v_i) - B^v_j - l^v_{Bj}) = 0 \qquad \forall v \in \mathcal{V}, (i,j) \in \hat{\mathcal{A}}^v | j \in \mathcal{N}^v, \tag{16}$$

$$l^v_{Bo(v)} = B^v_0 \qquad \forall v \in \mathcal{V}, \tag{17}$$

$$B^v_{\text{Min}} + \sum_{j\in\hat{\mathcal{N}}^v} B^v_{ij}(l^v_i) x^v_{ij} \le l^v_{Bi} \le \sum_{j\in\hat{\mathcal{N}}^v} B^v_{\text{Max}} x^v_{ij} \qquad \forall v \in \mathcal{V},\ i \in \mathcal{B}^v \cup \mathcal{N}^v \tag{18}$$

$$0 \le y^v_i \le (B^v_{\text{Max}} - B^v_{\text{Min}}) \sum_{j\in\hat{\mathcal{N}}^v} x^v_{ij} \qquad \forall v \in \mathcal{V},\ i \in \mathcal{B}^v \tag{19}$$

$$t^v_i + T^v_{i,N+i}(l^v_i) - t^v_{N+i} \le 0 \qquad \forall v \in \mathcal{V},\ i \in \mathcal{N}^v_P, \tag{20}$$

$$\sum_{j\in\hat{\mathcal{N}}^v} x^v_{ij} - \sum_{j\in\hat{\mathcal{N}}^v} x^v_{j,N+i} = 0 \qquad \forall v \in \mathcal{V},\ i \in \mathcal{N}^v_P, \tag{21}$$

$$x^v_{ij} \in \{0,1\}, \qquad \forall v \in V,\ (i,j) \in \hat{\mathcal{A}}^v. \tag{22}$$

The objective function (1) maximizes profit by subtracting all costs from revenues for serviced cargoes and adding the value of bonus bunker. The premium for bonus bunker is negative if the bunker level at the destination node is less than at the origin node. Constraints (2) and (3) ensure that all contract cargoes are carried by exactly one ship and that all spot cargoes are carried by at most one ship. Constraints (4) and (6) together with the flow conservation constraints in (5) ensure that each ship is assigned a schedule starting at the origin node and ending at the destination node. If a ship is idle for the entire planning period, the assigned schedule is simply represented by the arc $(o(v), d(v))$. Constraints (7) ensure that if the route for a ship v visits node i directly before node j, the service at node j cannot begin before service time at node i plus the service time at node i and travel time from node i to node j with ship v. Waiting time is allowed and, hence, the constraints have an inequality sign. Together with the time window constraints (8) they take care of the temporal aspect of the problem. If ship v does not visit node i, the service time t_i^v is artificial. Constraints (9), (10) and (11) ensure that the cargo load variables are correctly updated along the chosen route, adding the cargo quantity to the load variable if visiting a loading node, similarly subtracting the cargo quantity if visiting a discharge node and simply maintaining the previous load variable value if visiting a bunker node. In (12) the initial load condition for each ship is given since we assume that the ship is empty at the time it is available for service. Constraints (13) and (14) state intervals for the ship cargo capacity for loading and discharging nodes, respectively. Constraints (14) could be omitted as constraints (13) together with the precedence and coupling constraints (20) and (21) ensure that the upper bound will never be exceeded. Constraints (15)-(18) place similar restrictions on the bunker load variables: Constraints (15) and (16) ensure that the bunker load variables are updated correctly, constraints (17) give the initial bunker level for each ship while (18) give lower and upper bounds for the variables ensuring that a ship will never arrive at a port with less bunker than the safety level and will never carry more bunker than its bunker capacity allows. Constraints (19) restrict the bunker purchase amounts for each ship. Constraints (20) are precedence constraints ensuring that a cargo cannot be discharged before it has been picked up, i.e. node i must be visited before node $N+i$. Constraints (21) couple pickup and discharge nodes for each cargo together to ensure that the same ship will service both nodes. Finally, the flow variables are restricted to be binary in (22).

The research presented in this paper has been conducted in collaboration with the Danish shipping company Maersk Tankers A/S involved in, among other things, transportation of refined oil products worldwide. Based on their situation we focus on full ship loads. Note, however, that the mathematical model presented above is also valid for multiple cargo problems. In the case of full ship loads it is common industry practice to simply distinguish between a laden and a ballast ship, i.e. loaded or empty, rather than calculating the actual load, l_i^v. We will adopt this practice and can thereby use binary load variables, $l_i^v = l_{Ci}^v$, that is equal to 1 if the ship is laden and 0 if the ship is ballast. The time and bunker consumption functions as well as the cost functions thereby become dependent on a binary operator and ship capacity, V_{CAP}^v, and all cargo load quantities, Q_i, are set equal to 1. When considering full ship loads and not including bunkering, pickup and delivery of a single cargo must be performed directly after each other and, hence, the two tasks can be considered as just one task. This simplifies the model and constraints (9)-(14) and (20)-(21) can in that case be disregarded (as well as the bunker related constraints (15)-(19)). However, when including bunkering, bunker stops can be made in between pickup and delivery of a cargo so the two tasks cannot be aggregated into one. Instead the problem must be modeled similar to a multiple cargo problem as above.

4 Solution Method

The mixed integer programming model (1)-(22) could in theory be solved by commercial optimization software for non-linear problems. In practice, however, problem instances will be too large to achieve solutions in a reasonable amount of time. This section therefore describes a solution method tailored for the TSRSPBO.

In the mathematical programming model (1)-(22), constraints (4)-(22) are ship specific with no interaction between ships. They constitute a routing and scheduling problem for each ship where time windows, cargo and bunker capacity as well as bunker purchases are considered. We denote

these ship specific constraints *ship routing constraints* and further notice that the objective function also splits into separate terms for each ship. The only constraints linking the ships together are the so called *common constraints* in (2) and (3) which ensure that each contract cargo is carried by exactly one ship and that each spot cargo is carried by at most one ship. This suggests use of decomposition and column generation since it allows the complex and ship specific constraints, concerning the routing and scheduling, to be handled separately in *subproblems*, one for each ship. Only the common constraints remain in the *master problem* in which feasible ship schedules constitute the columns. This way the original problem is transformed into a master problem with a reduced number of constraints but with a potentially very large number of columns.

Often ship scheduling problems are so tightly constrained that it is possible to a priori generate all master problem columns. This is done by generating the optimal schedule for each feasible cargo set for each ship. Such an approach has been attempted in Brønmo et al. [2007]. However, as already mentioned, Brønmo et al. [2010] find it computationally advantageous to apply dynamic column generation even though a priori generation can be applied. The inclusion of bunker decisions in the scheduling process will further complicate the determination of an optimal schedule for a given cargo set and can, hence, make a priori generation very time consuming. In line with this, we apply dynamic column generation to solve the problem (see e.g. Desauliniers et al. [2005] for a general description or Christiansen et al. [2007] for a maritime version). Therefore, we initially consider only a subset of the master problem columns and iteratively add new columns that have the potential to improve the current solution. We find these columns by iteratively solving the subproblems, also called pricing problems.

4.1 The Master Problem

The common constraints (2) and (3) in combination with the objective function (1) constitute the master problem. They must, however, be expressed by new path flow variables corresponding to feasible ship schedules and constraints must be added to ensure that each ship is assigned exactly one schedule. We let $\mathcal{R}^v$ denote the set of all feasible schedules for ship v. Each cargo set can correspond to several feasible schedules as the order of cargoes in the schedule will correspond to different geographical routes. Schedules can also differ in bunker port calls, the amounts purchased at each bunker port and even in the timing of port calls. For a given set of cargoes there will be at least one profit maximizing schedule corresponding to the optimal bunkering strategy and the optimal timing of port calls. However, due to the subproblem solution method we might generate several different schedules for the same cargo set. We denote the profit of a schedule by p_r^v for $r \in \mathcal{R}^v$ and define a binary schedule variable λ_r^v that is equal to 1 if ship v is chosen to sail schedule r, and 0 otherwise. The profit p_r^v is calculated based on information from the underlying schedule, which holds all necessary information, i.e. the ship it is constructed for, the cargoes carried, the bunker ports visited as well as the bunker quantities purchased, and the timing of port calls during the schedule. Finally, we let a_{ir}^v denote the number of times ship v carries cargo i in schedule r.

The master problem is now given by the following path flow reformulation of the original arc flow model:

$$\max \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}^v} p_r^v \lambda_r^v \tag{23}$$

s.t.

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}^v} a_{ir}^v \lambda_r^v = 1, \qquad \forall i \in \mathcal{N}_C, \tag{24}$$

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}^v} a_{ir}^v \lambda_r^v \leq 1, \qquad \forall i \in \mathcal{N}_O, \tag{25}$$

$$\sum_{r \in \mathcal{R}^v} \lambda_r^v = 1, \qquad \forall v \in \mathcal{V}, \tag{26}$$

$$\lambda_r^v \in \{0,1\}, \qquad \forall v \in \mathcal{V},\ r \in \mathcal{R}^v. \tag{27}$$

The above model is based on all feasible schedules but it is not necessary to include all of them. Instead, column generation is applied to dynamically generate them as needed. This process

begins with the solution of the *restricted master problem* (RMP) which is the linear relaxation of the original master problem (23)-(27) but with only a subset of the columns included. Iteratively we then generate new promising columns by solving the subproblems.

4.2 The Subproblem - Generation of promising schedules

Constraints (4)-(22) split into one independent subproblem for each ship. Since these are all essentially the same problem, we simply consider the generic subproblem for ship v and refer to 'the subproblem'. Note though the interdependence between the subproblems due to the common constraints. The ship routing constraints in the subproblem ensure that any solution is a feasible schedule for ship v and the objective ensures that only schedules with the potential to improve the current solution of the RMP are generated. This, in turn, means finding schedules with positive reduced costs in the current solution of the RMP, i.e. finding columns r with $p_r - \sigma^T a_r$, where σ is the dual vector of the current solution of the RMP. Let u_i be the dual variables for constraints (24) and (25) and w_v be the dual for constraint (26). Next, let $\sigma_i = u_i$ for all $i \in \mathcal{N}_P$, $\sigma_{o(v)} = w_v$ corresponding to the origin node and $\sigma_i = 0$ for all other i. Since we consider the generic subproblem we can drop the superscript v and the subproblem is then given by:

$$\max \sum_{i \in \mathcal{N}_P} R_i \Big(\sum_{j \in \hat{\mathcal{N}}} x_{ij} \Big) - \sum_{(i,j) \in \hat{\mathcal{A}}} \big(C_{ij}(l_i) + \sigma_i \big) x_{ij} - \sum_{k \in \mathcal{B}} y_k P_k + P(l_{Bd} - B_{\text{Min}}) \tag{28}$$

s.t.

$$(4) - (22). \tag{29}$$

The subproblem finds the maximum reduced cost feasible schedule with respect to the current dual values. If this schedule has a positive reduced cost it will be represented by a new column in the RMP. The subproblem can be modeled as a resource constrained shortest path problem and is $\mathcal{NP}$-hard since it is a generalization of the shortest path problem with time windows which is itself $\mathcal{NP}$-hard (see e.g. Desrosiers et al. [1995]). We therefore devote Section 4.4 to an efficient solution method for the subproblem.

4.3 Full Column Generation Scheme

The full column generation scheme is an iterative process starting from the RMP with only a small initial column set. To ensure feasibility of the initial problem, we include a dummy column for each contract cargo. Each dummy column corresponds to an artificial ship carrying exactly one contract cargo. The revenue from each dummy schedule is $-M$, where M is a large constant. Feasibility with respect to the generalized upper bound constraints (26) must also be ensured, i.e. each ship must be assigned a schedule. Therefore, we include an empty schedule with 0 profit for each ship corresponding to the ship being idle for the entire planning horizon, and leave the corresponding $(o(v), d(v))$ out of the subproblem networks.

Once the RMP has been solved, the optimal dual solution values are transferred to each of the subproblems which are then solved to obtain new schedules. All new schedules with positive reduced cost are transformed into columns which are added to the RMP. The updated RMP is resolved to obtain new dual values that can again be transferred to the subproblems. This process of iterating between the master problem and the subproblems continues until no promising columns can be found, i.e. until no schedules have positive reduced costs.

Since all the intricate and nonlinear constraints and costs are transferred to the subproblems, the master problem can most often be solved by commercial linear programming software. As each subproblem must be solved a potentially great number of times to obtain all necessary columns, a fast solution method for the subproblem is vital for the effectiveness of the column generation scheme. Since the subproblem considered here is $\mathcal{NP}$-hard, solving it can be very time consuming and a choice between heuristics and optimization must be made depending on the desired solution quality and computation time. However, often it is only slightly more time consuming to find several schedules at a time, and, hence, let each subproblem generate several columns in each iteration. This can speed up the solution process considerably, and the subproblem solution method presented in the next section exploits this fact.

Once the column generation process terminates, an optimal solution to the linear relaxation of the full master problem is obtained. In order to ensure an optimal integral solution, the column generation scheme must be embedded in a Branch & Bound search, resulting in a Branch & Price algorithm. In our computational studies we have, however, encountered relatively few fractional solutions and for these fractional solutions, the integrality gap was acceptable. Therefore, we have not implemented a Branch & Bound algorithm. Instead, integrality has been enforced by the simple, but non-optimal, approach of solving the integer version of the RMP once column generation terminated. In Section 7 we verify the quality of these forced integer solutions by comparing them to their corresponding upper bounds obtained from the fractional solutions.

4.4 Solving the subproblem

Shortest path problems with resource constraints (SPPRC) are often encountered in both land and air based transportation, e.g. as a subproblem in vehicle routing and crew rostering. The problem is most often solved by dynamic programming algorithms on the underlying networks and we will also use this approach. We refer the reader to Desaulniers et al. [1998], Irnich and Desaulniers [2005] and Irnich [2008] for a thorough introduction to the SPPRC, dynamic programming algorithms and concepts such as labels, resource extension functions and dominance functions.

The overall idea behind these algorithms is to construct partial paths, which in this case corresponds to partial schedules, from the origin node, $o(v)$, to all its successor nodes and iteratively extend these partial schedules to all possible successor nodes until no schedule can be extended. At this point, any schedule that reaches the destination node $d(v)$ represents a resource feasible solution. Throughout the algorithm each partial schedule is represented by a label and two schedules arriving at the same node can be compared by defining a partial order relation between labels. This partial order allows us to determine if one label dominates another that can, hence, be discarded. This dominance concept ensures that only the best schedules, i.e. Pareto optimal, are kept during the iterative process of the algorithm as only they can contribute to the optimal schedule. The algorithm relies heavily on the domination procedure to efficiently eliminate dominated labels, thereby reducing the potentially huge label set.

The reduced cost objective function (28), aside from the constant node costs and the load dependent arc costs, also has costs that are linear in the continuous bunker purchase variables y associated with the bunker nodes and in the bunker load variable at the destination node. Ioachim et al. [1998] present an optimal dynamic programming algorithm to solve the shortest path problem with time windows and additional linear costs on the node service start times. Similarly, Christiansen and Nygreen [2005] consider a maritime transportation problem with linear costs in the node arrival times. This inclusion of linear node costs means that the common time-constrained shortest path problem labels with constant cost are replaced by linear cost functions over time intervals and that the standard list of non dominated labels with constant cost are replaced by a piecewise linear cost function obtained from minimizing over the arcs into each node.

A similar approach could be considered here to handle the continuous bunker purchase variables. Note though, that the linear node costs in our problem concern variables that are associated with a single node rather than in the common variable time as in Ioachim et al. [1998] and Christiansen and Nygreen [2005]. We would therefore have to aggregate the labels into a piecewise linear function of a common variable such as the bunker inventory onboard the ship on arrival at the node the schedule ends in. However, the research in this paper is on a more tactical level than an operational one. We are looking for a guide line on where to bunker and roughly how much to bunker at each bunker stop. The operational planning problem of exactly how many tons of bunker with decimal accuracy is not relevant in our setting where decisions are based on bunker price forecasts rather than actual prices. Therefore, we have chosen to avoid the added algorithmic complexity just to achieve decimal accuracy. Instead, we will discretize the bunker purchase variables when solving the subproblem and apply a standard label setting algorithm. We apply the algorithm to extended cargo-bunker ship networks with discrete bunker purchases. This way there will be several nodes in the subproblem for the same bunker option - one for each possible bunker purchase quantity.

When discretizing the bunker purchase variables an obvious concern is how to do this in a manner that does not take the heuristic solution too far away from the optimal solution. We therefore note that logically the optimal decision at each bunker stop is to either fill up the tank

or to purchase just enough bunker to allow the ship to sail to the next bunker stop. In advance, we cannot now how much bunker is required to get to the next bunker stop but we can obviously fill up the tank. When tuning and testing the devised solution method in Section 6 and Section 7 we see that when constructing bunker schedules, the majority of bunker stops actually correspond to filling up the tank to its capacity. As we want to retain this optimal decision amongst the possible purchase quantities, we let each bunker node correspond to the situation of filling up the tank to a certain inventory level rather than purchasing a specific amount of bunker. Note that using a mix of these two types of bunker purchases can almost aggregate some bunker nodes, e.g. if the 'fill up tank' node corresponds to bunkering 833 tons and the fixed amount node is 800 tons. Therefore, we only use 'fill up to' nodes so that the purchase quantities span as much as possible of the feasible interval of possible purchase quantities. Each original bunker node in the subproblem is replaced by a set of bunker nodes all associated with the original bunker node but each corresponding to a different 'fill up to' level. If any arcs connecting these new nodes to the rest of the network are infeasible with respect to bunker consumption and safety levels, they can of course be removed. Note, that the actual amount of bunker purchased is not fixed as it depends on the bunker inventory on arrival at the node. Instead, the bunker purchase contribution to the reduced costs must be calculated dynamically as bunker inventory levels are iteratively updated. The premium for bonus bunker must also be calculated dynamically.

Each ship has its own bunker tank capacity and safety level and these, in turn, describe the ship specific interval of feasible bunker purchase quantities. We divide this bunker purchase interval, $[B_{Min}, B_{Max}]$, into L discrete bunker purchase quantities, where L is a parameter of the algorithm that we tune in Section 6. Note that this parameter could be different for each ship but as the tank capacities do not vary too much in size on the fleet we consider, we have chosen to use the same parameter for all ships. Dividing $(B_{Max} - B_{Min})$ into L intervals and rounding the result down to the nearest 25 tons yields the refinement level q. Each original bunker node in the subproblem is now replaced by L nodes with purchase quantity levels B_{Max}, $B_{Max} - q \ldots$, $B_{Max} - (L-1)q$. If $L = 1$ the only option is to fill up the tank to its maximum capacity.

With the discretization of bunker purchases, the notion of bonus bunker becomes a vital part of the solution procedure. If this premium for unused bunker is not included in the model, it becomes more important to find combinations of bunker purchase quantities that let ships finish their schedules with empty tanks than to find cheap bunker options. I.e. the effect of price changes on the optimal bunker plan is almost non existent.

Before setting up the networks, we make an assumption that reduces the potential size of the networks: A ship makes at most one bunker stop in between cargo stops. Data from the collaborating tramp operator on distances, port costs, fleet bunker consumption and fleet bunker tank capacity shows that this assumption is indeed very reasonable.

When setting up the ship specific networks we utilize the assumption of full ship loads. Aside from intermediate bunker stops, this assumption means that we always know where a loaded ship is going: to the discharge port of the onboard cargo. We therefore set up the networks by further duplicating all the bunker nodes, i.e. $K \cdot L$ with $K = |\mathcal{B}^v|$, as many times as there are potential time feasible arcs in the standard network, $(\mathcal{N}^v, \mathcal{A}^v)$, excluding arcs leading to the destination node. Note that 'potential time feasible' means that any arc connecting nodes in $\mathcal{N}^v \setminus d(v)$, and which is feasible with respect to time, is considered here. A 'potential time feasible' arc does not have to belong to $\mathcal{A}^v$ as it can be infeasible with respect to bunker constraints. Figure 1-3 illustrate this extension process of the standard network for a small example with 2 cargoes and only one bunker option that has been discretized into just one node, i.e. $L = 1$. Figure 1 shows the standard network, Figure 2 shows the extended cargo-bunker network that utilizes the assumption of full ship loads while Figure 3 shows the normal cargo-bunker network setup that would be used in case of multiple cargoes onboard. Note the double headed arrows in Figure 3 which have just been aggregated for the sake of simplicity in this figure. Nodes L_i and D_i correspond, respectively, to pickup and discharge of cargo i while node B correspond to the bunker node.

Setting up the networks in such a manner means that all paths in the networks respect the cargo capacity and precedence constraints as they are implicitly considered in the network setup. Furthermore, each arc in the network now corresponds to either a laden or a ballast ship and, hence, the load dependent time and bunker consumption as well as costs can now be considered fixed and calculated in advance. In total, this network setup allows us to disregard the onboard

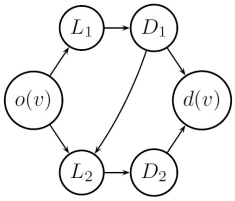

Figure 1: Standard NW

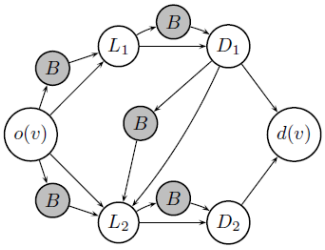

Figure 2: Extended CB NW

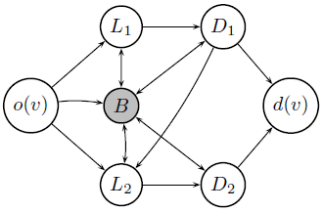

Figure 3: Normal CB NW

cargo as a resource and this reduction in resources simplifies both the labeling, domination and path extension procedures considerably. All these advantages do, however, come at the expense of an increase in the potential node and arc count of $(N^2 + N - 1)K \cdot L$ and $(2N^2 - 2N - 1)K \cdot L$, respectively, compared to the normal cargo-bunker network illustrated in Figure 3. A significant number of these extra nodes and arcs can, however, be removed by preprocessing, especially with respect to time windows. A further reduction in the node and arc count can be achieved by noting that some of these new bunker nodes can be aggregated into one without sacrificing the reduction in resources. First, all bunker nodes that are successors to the origin node and correspond to the same bunker option and purchase level can be aggregated. Secondly, all bunker nodes that are successors to a single discharge node and predecessors to other pickup nodes and correspond to the same bunker option and purchase level can be aggregated. This reduces the potential size of the network by $(N^2 - N - 1)K \cdot L$ nodes and also $(N^2 - N - 1)K \cdot L$ arcs compared to the extended cargo-bunker network illustrated in Figure 2. This gives a potential network size of $2N + 2 + (2N + 1)K \cdot L$ nodes and $N^2 + 2N + (N^2 + 3N + 1)K \cdot L$ arcs. We illustrate this final network setup in Figure 4 for the same example as above.

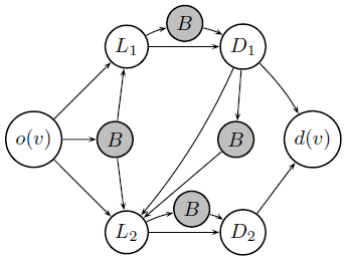

Figure 4: Reduced extended CB NW

During network construction, standard preprocessing techniques are applied to tighten time windows and, in turn, reduce the number of arcs, see e.g. Desrosiers et al. [1995]. Each network now have cargo nodes, bunker nodes and an origin and destination node. Each node has an associated time window, an associated port and for bunker nodes also a bunker price. Each node also has a bunker window holding the minimum and maximum level of bunker allowed onboard a ship on arrival. For cargo nodes and the destination node this window is $[B^v_{Min}, B^v_{Max}]$ and for the origin node it is $[B^v_0, B^v_0]$. For the bunker nodes, the window becomes $[B^v_{Min}, F]$, where F is the 'fill up to' level at the corresponding bunker node. Each arc in the network now has a constant time and bunker consumption as well as cost and we denote these by T_{ij}, B_{ij} and C_{ij}, respectively. We abuse notation slightly by now letting B_{ij} denote bunker consumption corresponding to both traveling from i to j as well as the consumption from port operations at node i (as opposed to $B^v_{ij}(l^v_i)$ that did not include port operations at node i). Note that C_{ij} still does not include bunker purchases or the bonus bunker premium as these will be added dynamically during the algorithm.

Since we solve the subproblem as a shortest path problem, we assign the negative of the fixed cost part of the reduced costs to each arc (i, j), and denote this by $\hat{C}_{ij}$. I.e.

$$\hat{C}_{ij} = C_{ij} + \sigma_i - R_i \qquad \forall (i,j) \in \hat{\mathcal{A}}, \tag{30}$$

where $R_i = 0$ for $i \notin \mathcal{N}_P$. The remaining part of the reduced cost expression in (28), i.e. the node

costs for bunker purchases and node premiums for bonus bunker, must be added dynamically as partial schedules are extended and bunker purchase amounts are determined.

Time windows on bunker nodes are, as mentioned, so narrow that a ship can never visit the same bunker node twice. In our data sets, the time windows for cargo loading are also tight enough that, in combination with the long voyage lengths, there do not exist any time feasible cycles in the networks. We model the subproblem on an acyclic graph, and if cycles do in fact exist, nodes with wide time windows must be split into several duplicate nodes with smaller time windows. Although this produces an acyclic network, it does not ensure that cargoes are not lifted several times in one schedule as this simply corresponds to visiting several of the duplicate nodes for the same cargo. To avoid this, the label for a path must include information on nodes visited so far. This would, however, be computationally intractable due to the labeling process and the domination procedure. Therefore, we relax the subproblem to allow non elementary paths and refrain from keeping track of the nodes previously visited. Hence, routes can be produced where a cargo is picked up more than once, i.e. have $a_{ir} > 1$ for some i. Such a column will not be added to the master problem.

For a schedule s we denote by $\bar{C}(s)$ the negative of the reduced cost for the schedule, i.e. the sum of the arc and node costs where the arc costs are $\sum_{(i,j)\in s} \hat{C}_{ij}$. The node costs depends on the bunker purchases during the schedule and also the premium for bonus bunker if the schedule ends at the destination node. These are schedule dependent and will therefore be calculated dynamically in the algorithm for each specific schedule. To each schedule s_i ending in node i we associate a label $\mathcal{L}(s_i) = (\bar{C}(s_i), T(s_i), B(s_i))$ where $T(s_i)$ and $B(s_i)$ denote, respectively, the arrival time at node i and the bunker inventory level on arrival at node i on schedule s_i.

The domination procedure must ensure that all schedules that are not Pareto optimal, and only these, are discarded during the algorithm. For this, we note that a schedule s_i ending at node i dominates another schedule s_i' also ending at node i if and only if $\mathcal{L}(s_i) \neq \mathcal{L}(s_i')$, $\bar{C}(s_i) \leq \bar{C}(s_i')$, $T(s_i) \leq T(s_i')$ (since there is no cost for waiting) and $B(s_i) \geq B(s_i')$. Any two schedules arriving at the same node can now be compared according to this partial order relation and dominated schedules and labels can be discarded.

When extending a partial schedule, the resource extension functions ensure correct extension of the label associated with the schedule. For the time resource this means that $T(s_j) = \max\{T_{MNj}, T(s_i) + T_{ij}\}$ and that the extension is only allowed if $T(s_i) + T_{ij} \leq T_{Mxj}$. When extending to a bunker node, the bunker resource is updated as $B(s_j) = B_{MXj}$ with an associated dynamically calculated purchase quantity $y_j = B_{MXj} - (B(s_i) - B_{ij})$. Note that this updates the bunker inventory immediately on arrival at the node and remember that B_{MXj} is defined as the 'fill up to' level for bunker nodes. I.e. a ship filling up its bunker tank to e.g. 1200 tons might actually leave the node with 1199 tons as a small amount of bunker is consumed while bunkering. If j is a cargo related node, we have $B(s_j) = B(s_i) - B_{ij}$. For the destination node, d, we get $B(s_d) = B_0$, with an associated dynamically calculated bonus bunker amount, y_d, given by $B(s_i) - B_{id} - B_0$. With this setup, s_i can only be extended to node j if $B(s_i) - B_{ij} \geq B_{MNj}$ and $B(s_i) - B_{ij} \leq B_{MXj}$. The latter requirement ensures that a schedule with arrival bunker inventory higher than the 'fill up to' level at a bunker node will not visit such a node. Finally, the negative of the reduced costs are updated as follows:

$$\bar{C}(s_j) = \bar{C}(s_i) + \hat{C}_{ij} + y_j P_j, \qquad \forall j \in \mathcal{B}, \tag{31}$$

$$\bar{C}(s_j) = \bar{C}(s_i) + \hat{C}_{ij}, \qquad \forall j \in \mathcal{N}, \tag{32}$$

$$\bar{C}(s_d) = \bar{C}(s_i) + \hat{C}_{id} - y_d \cdot P, \tag{33}$$

Using labels, resource extension functions and domination procedure as described above, we apply dynamic programming to the subproblem with nodes sorted topologically according to time.

When the algorithm terminates, several resource feasible and Pareto optimal schedules might exist. We add all schedules with positive reduced cost, i.e. $\bar{C}(p) < 0$, to the master problem. Due to the reselling of bonus bunker, all schedules will have the same amount of bunker at the end, namely the initial inventory level of the ship. Schedules can, however, differ in both reduced costs and time. Therefore, we can have multiple columns corresponding to the same cargo set in the master problem if they correspond to different end times, e.g. due to differences in bunker plans.

Finally, it should be noted that due to the discretization of bunker purchases and the assumption of at most one bunkering in between cargo stops, the subproblem solution method described above

is heuristic. To ensure an optimal solution to the master problem, the subproblems should be solved to optimality once the heuristic approach fails to find schedules with positive reduced costs. As previously discussed, the planning problem considered here is of a more tactical nature and, hence, an optimal continuous solution is beyond the scope of this research. We do, however, rerun the dynamic programming algorithm with an increased number of possible purchase quantities, i.e. an increased value of L, for the fixed cargo routes found by the initial optimization of the master problem. We discuss this further when tuning the algorithm in Section 6.

5 Problem Instance Generators

In order to both tune and test the devised algorithm thoroughly, we have developed instance generators that independently generates cargoes and bunker prices. These instance generators are based on industry data from the collaborating tramp operator. Although this operator operates world wide, the cargoes naturally divide into two groups traveling within two separate parts of the world with only 5% of cargoes traveling between them. We therefore limit our analysis to one of these cargo groups, namely the one responsible for almost 70% of the overall cargoes. We have excluded some remote regions that generate very little, if any, demand. This leaves us with a cargo area covering the Mediterranean, the North-West part of Europe, the East Coast of Canada and the US, the Mexican Gulf and the Caribbean Sea. We have selected 38 ports that are representative for the ports in this area. Both generators therefore assume 38 ports and each port has some associated ship dependent port costs. For all problem instances the fleet is the same and consists of 7 ships of varying size and other characteristics, e.g. speed, bunker consumption etc.

For each cargo, the cargo generator randomly selects a pickup port from a probability distribution of cargo pickup ports. Once the pickup port is known, there is a specific discharge distribution related to this pickup port from which a discharge port is randomly drawn. A cargo quantity is randomly selected in a user defined interval. For our analysis, the quantities are randomly chosen between 60-90% of ship sizes. Based on this quantity as well as the distance between pickup port and discharge port and their costs, a reasonable revenue for transporting the cargo is randomly calculated. Also based on user defined intervals, time windows for both pickup and discharge as well as the service time for loading and unloading are randomly calculated. We have used time windows with a length of minimum 72 hours and maximum 120 hours. Finally, the cargo is randomly selected to be either a spot cargo or a contract cargo.

The bunker price generator randomly generates a price quote for each of the 38 ports for a number of consecutive time periods determined by the user. E.g. if a period is determined to be 3 days and the user asks for 20 bunker options, 20 price quotes will be generated for each of the 38 ports and each of these prices are given a time window of 3 days. For the 38 prices of the last time period in the planning period, an average is calculated to use for bonus bunker. To generate the actual prices, the 38 ports are divided into regions and each port is randomly selected to belong to one of the price classes cheap, average and expensive. For each region, reasonable bunker price intervals corresponding to a cheap port, an average port and an expensive port at the beginning of the planning period are given as parameters to the generator. Each port is assigned a start price, e.g. a price for the first 3 days, by randomly picking a price in the interval that corresponds to the specific region and price class of the port. In order to generate prices for the remaining time periods a world trend is randomly generated that is valid for all regions and ports. This world trend simply defines whether the price goes up or down from one time period to the next. For each port the remaining bunker prices are now determined by using the start price of the port and then raising or lowering the price from time period to time period following the world trend. The actual amount it is raised or lowered with is determined randomly for each port for each time period from an interval defined by user input. In our analysis we have used an interval of 0-5%.

6 Parameter Tuning

The number of possible purchase levels, i.e. the parameter L, must be tuned before running the algorithm. Obviously, the more levels we allow the more of the underlying feasible bunker interval we span, however at a cost of computation time.

We have generated 18 problem instances for tuning using the instance generators described in Section 5. All problem instances have the same fleet of 7 ships and use 38 ports worldwide. Three cargo instances have been generated containing 30 cargoes with their loading time windows distributed over a time horizon of 30 days. Note that the planning period continues after these 30 days as cargoes must of course also be discharged. Three bunker price instances have been generated with 14 weekly bunker options for each port corresponding to a time horizon that just contains the latest possible discharge time plus the time to discharge. Combining each cargo instance with each of the corresponding bunker instances yields nine instances with this combination of data and we denote them C30/PH30/B14 instances. Another three cargo instances have been generated with 50 cargoes over a 60 days pickup planning horizon. Three bunker price instances have been generated for these cargo sets but now with 19 bunker options for each port. Again, we get nine instances by combining each cargo instance with each of the bunker instances. We denote these bigger instances by C50/PH60/B19. All cargoes are defined to be spot cargoes.

On each of the problem instances we have run the algorithm with varying number of purchase levels, namely L varying from one to ten, and report the key values in Table 1. Each entry corresponds to the average over the nine problem instances of the specific instance type for the stated setting of L. The key values reported are: the percentage increase in the objective function value compared to the $L = 1$ case (Obj.), the total running time in CPU seconds ($\text{CPU}_{\text{Total}}$), the CPU seconds for solving the subproblems (CPU_{Sub}), and, finally, the percentage of all bunker stops that filled up the ship's bunker tank to its maximum capacity (Filled). This number is interesting as it shows that relatively few bunker stops use 'fill up to' levels lower than tank capacity and, hence, the actual discretization of this interval is less important.

	C30/PH30/B14				C50/PH60/B19			
	Obj.	$\text{CPU}_{\text{Total}}$	CPU_{Sub}	Filled	Obj.	$\text{CPU}_{\text{Total}}$	CPU_{Sub}	Filled
$L = 1$	-	5.8	5.5	100.0	-	31.6	30.9	100.0
$L = 2$	0.91	14.0	13.6	78.6	0.53	77.2	76.1	77.3
$L = 3$	1.08	23.9	23.4	74.4	0.58	116.5	115.1	78.8
$L = 4$	1.18	35.2	34.6	73.1	0.69	186.9	185.2	72.4
$L = 5$	1.25	48.5	47.8	75.2	0.74	276.9	274.7	70.2
$L = 6$	1.32	71.7	70.9	74.4	0.81	360.6	358.0	67.9
$L = 7$	1.34	86.7	85.7	73.1	0.86	495.5	492.5	67.3
$L = 8$	1.35	111.1	110.0	72.5	0.88	540.9	537.7	64.6
$L = 9$	1.37	144.4	143.2	72.5	0.86	755.0	751.4	64.9
$L = 10$	1.40	157.2	155.9	73.1	0.89	861.5	857.6	64.3

Table 1: Tuning results for increasing L values.

We see from Table 1 that increasing L yields an objective function value increase of only 0.53-1.4% and that the increase is largest when going from $L = 1$ to $L = 2$. Note also that the objective function value does in one case drop when increasing L. This demonstrates the heuristic nature of the algorithm due to the discretization. Furthermore, the increase in computation time is considerable as L is increased and this is almost only due to the increase in solution time for the subproblems. Running the algorithm with high L values is therefore computationally undesirable. Increasing L gradually during the algorithm as the optimum is approached will also be very time consuming. Even resolving the subproblems in each iteration with an increased value of L for each fixed cargo route found by the shortest path solver, i.e. each Pareto optimal schedule (or the best of them), will be computationally expensive. Instead we have chosen to investigate the effect of simply increasing L for the fixed cargo sets found for each ship in the final solution to the master problem. We rerun the algorithm with $L = 17$ as this is the lowest value that yields a refinement of 25-50 mts between purchase quantities for all ships. For our tactical approach this level of refinement is sufficient and mimics a continuous solve.

Table 2 shows the key values for rerunning the algorithm on all 18 instances again for increasing values of L but this time finishing the algorithm by solving the bunker optimization problem with $L = 17$ for the fixed cargo sets determined by the final solution to the master problem. If the original solution from using the low L-value is better than when rerunning the algorithm with

$L = 17$ for fixed routes, we naturally use the original solution rather than the solution from rerunning with $L = 17$. Note that the objective column (Obj.) again contains the percentage increase in profit compared to the $L = 1$ case for the unrefined algorithm, i.e. the one used in Table 1.

	9 instances: C30/PH30/B14			9 instances: C50/PH60/B19		
	Obj.	$\text{CPU}_{\text{Total}}$	Filled	Obj.	$\text{CPU}_{\text{Total}}$	Filled
$L = 1$	1.33	14.6	76.6	0.70	61.5	66.0
$L = 2$	1.46	21.3	73.8	0.95	108.7	65.5
$L = 3$	1.45	31.7	73.5	0.94	145.8	65.2
$L = 4$	1.44	42.8	73.6	0.97	218.6	64.4
$L = 5$	1.43	56.2	74.4	0.93	307.3	65.7
$L = 6$	1.46	79.3	72.5	0.97	390.2	63.5
$L = 7$	1.46	94.0	75.0	0.96	526.0	64.5
$L = 8$	1.47	118.7	73.8	0.97	572.8	64.3
$L = 9$	1.47	151.9	73.3	0.94	785.3	63.6
$L = 10$	1.47	164.8	73.8	0.97	892.1	64.1

Table 2: Tuning results for algorithm that reoptimizes with $L = 17$ for increasing L values.

As before, we see that increasing the value of L yields almost no, if any, increase in profit and yet the computation time increases rapidly.

In Figure 5 and Figure 6 we illustrate these findings for the C30/PH30/B14 and the C50/PH60/B19 instances, respectively. Each figure shows a plot of the percentage increase in objective function value and the CPU seconds both as functions of L for the standard version of the algorithm and for the refined version using a resolve on fixed cargo sets with $L = 17$.

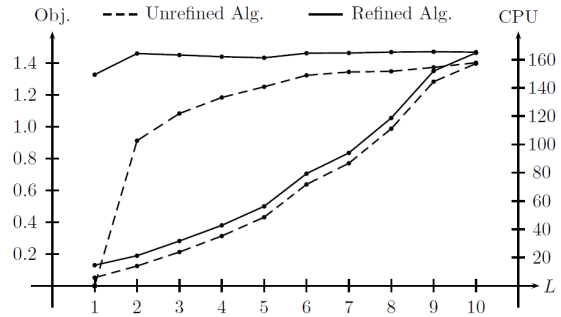

Figure 5: Tuning for small instances

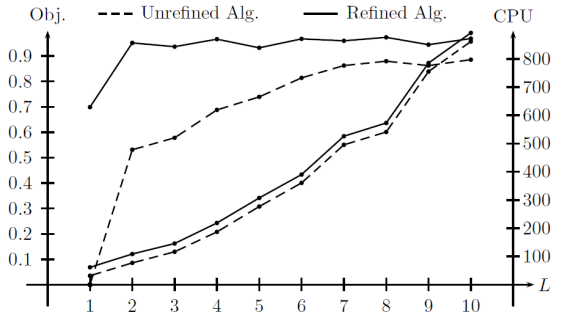

Figure 6: Tuning for large instances

As can be seen from the above figures, the refined algorithm with $L = 2$ afterwards increased to $L = 17$ yields almost the best objective function values of all settings and at almost no increase in computation time. Increasing the initial L-value above 2 achieves at best an insignificant objective improvement of only 0.01% and this is at great computational expense. We therefore use the refined algorithm with an initial value of $L = 2$ when testing the algorithm in the next section.

7 Computational Results

In order to explore the benefits of integrating bunker planning in the routing and scheduling phase, we compare the devised solution method with the standard sequential approach where routes and schedules are planned with no consideration to actual bunkering. When planning routes and schedules in this standard approach, bunker consumption is accounted at the average of all bunker prices valid at the time of planning and no actual bunker stops are planned, meaning that no time is scheduled for bunkering and no bunker port costs are incurred. Each optimal schedule from this process now assigns a given cargo set to each ship and a bunker plan must be created that

respects this cargo assignment to ships. We find the optimal bunker plan by fixing cargoes to ships according to this cargo assignment, and then run the bunker algorithm with $L = 17$.

When testing the devised solution method, we consider the same fixed fleet of 7 ships as when tuning in Section 6 and also use the same 38 ports for all test instances. We have used the instance generators described in Section 5 to generate 25 cargo instances. With a pickup time horizon of 30 days we have five sets with 30 cargoes, five sets with 40 cargoes and five sets with 50 cargoes. For the 60 days horizon we have five sets with 50 cargoes and five sets with 60 cargoes. Again, all cargoes are defined to be spot cargoes. We also generated problem instances with a planning horizon of 90 days but when testing on these instances we found that the sequential approach in 2 out of 3 cases produced a solution that was infeasible with respect to bunkering. Therefore, we could not compare the two methods on these cases and do not report them here. It should be noted that bunkering becomes more relevant the longer the planning horizon we consider as ships must travel more, but at the same time the assumption of valid price forecasts becomes more unrealistic. The bunker price generator has been used to generate six bunker instances: Three price instances for the 30 days planning horizon with 14 weekly bunker options per port, i.e. a total of 532 bunker options, and three price sets for the 60 days horizon with 19 bunker options per port, i.e. 722 bunker options. These cargo and price instances can be combined to a total of 75 problem instances. Table 3 gives an overview of these problem instances of varying size and complexity.

Ships	7	7	7	7	7
Ports	38	38	38	38	38
Cargoes	30	40	50	50	60
Pickup Time Horizon (days)	30	30	30	60	60
Bunker options per port	14	14	14	19	19
Bunker options in total	532	532	532	722	722
Number of instances	15	15	15	15	15

Table 3: Problem instance overview.

We use the same notation as in Section 6 and denote a problem instance with e.g. 40 cargoes over a pickup horizon of 30 days with 14 bunker options per port by C40/PH30/B14.

On each of these 75 problem instances we have run both the standard sequential approach described above and the integrated approach defined by the refined bunker algorithm described in Section 4 and Section 6 with $L = 2$ afterwards increased to $L = 17$. All computational experiments were performed on a PC with 4.0 GB RAM and an Intel(R) Core(TM)2 Duo CPU P8600, 2.4 GHz processor under a 64 bit Windows 7. Both algorithms where entirely developed in C++ using Cplex 12.4 with default settings to solve the master problem.

Table 4 summarizes some key values for the refined bunker algorithm. Each line corresponds to the average key values over the 15 problem instances of the corresponding problem type given by the entry in the left most column. We do not report the objective function value but return to that when comparing with the sequential approach. The key values reported are, respectively, the percentage gap from the forced integer solution to the LP solution (Gap), CPU seconds for the whole algorithm ($\text{CPU}_{\text{Total}}$), CPU seconds for reoptimizing bunker with $L = 17$ for fixed routes (CPU_{17}), CPU seconds for solving all subproblems in the column generation phase with $L = 2$ (CPU_{Sub}), the number of columns generated (Cols.) and the number of calls to the subproblems (Subs) (i.e. number of iterations) in the column generation procedure with $L = 2$, the percentage of all bunker stops that corresponded to filling up to tank capacity (Filled), and finally, some statistics on price sensitivity (PS(Av,Max)). As each cargo instance is run with three different price instances we can get an idea of how sensitive the method is to changes in prices. For this we consider the differences in carried cargoes from two solutions derived from the same cargo instance but from different price instances. If one solution carries x more cargoes than the other solution, we define the *cargo difference* to be equal to x. Two solutions carrying the same number of cargoes do not necessarily carry the *same* cargoes and we increase the cargo difference count by one for each difference in carried cargoes when comparing the two solutions. As a small example, imagine that one solution carries cargoes 1, 2 and 3 while another solution carries cargoes 1, 3, 4 and 5. Such

a solution would correspond to one extra cargo *and* one different cargo and we would therefore define the cargo difference between these two solution to be equal to two. In the price sensitivity column (denoted PS) we report the average and the maximum cargo difference when comparing solutions derived from the same cargo instance.

	Gap	CPU_{Total}	CPU_{L17}	CPU_{Sub}	Cols	Subs	Filled	PS(Av,Max)
C30/PH30/B14	0.00	17.1	6.8	9.9	205	6	70.0	(1.2 , 2)
C40/PH30/B14	-	27.3	10.5	16.3	248	6	75.6	(0.9 , 2)
C50/PH30/B14	0.22	44.8	14.0	30.1	287	7	71.8	(2.2 , 5)
C50/PH60/B19	0.26	123.9	33.9	88.8	453	11	65.3	(3.8 , 8)
C60/PH60/B19	0.07	167.4	37.9	128.1	513	11	64.5	(3.9 , 8)

Table 4: Key values for the refined bunker algorithm.

Out of the 75 test instances, we obtained fractional solutions from only 19 instances and from Table 4 we note that the integrality gap is relatively small for these fractional occurrences. Aside from justifying our non optimal integer approach this also suggests that the ships are not competing for the cargoes. This is probably because the fleet operates in a very large part of the world, and, hence, ports are spread over vast distances. For a given cargo, chances are that only one available ship is close enough for it to be profitable to carry the cargo. Vice versa, for a given ship, there are only a few cargoes that are both reachable with respect to time but also profitable. From Table 4 we also see that the majority of bunker stops correspond to filling up to tank capacity. Finally, from the price sensitivity column we see that the optimal solution is indeed affected by changes in prices. For the larger instances, two solutions from the same cargo instance can differ by as much as 8 cargoes making an accurate price forecast very important.

Before considering the sequential approach we first present some network statistics in Table 5 for the bunker optimization with $L = 17$. In the first part of the table we report statistics on the actual network sizes: The number of nodes in the average subproblem network (Nodes), the number of these that were bunker nodes (bNodes), the arcs in the average network (Arcs) and, finally, the number of these that where bunker arcs (bArcs). In the second part of the table we report the potential network sizes of, respectively, the aggregated (agNodes and agArcs) and the extended (extNodes and extArcs) cargo-bunker networks as stated in Section 4.4.

	Actual Network Size				Potential Network Size			
	Nodes	bNodes	Arcs	bArcs	agNodes	agArcs	extNodes	extArcs
C30/PH30/B14	19,362	19,303	39,423	39,252	64,966	1,055,384	989,582	1,980,000
C40/PH30/B14	27,612	27,534	56,974	56,688	86,266	1,832,824	1,745,042	3,491,600
C50/PH30/B14	35,833	35,736	77,165	76,735	107,566	2,823,264	2,713,302	5,429,000
C50/PH60/B14	49,899	49,802	150,177	149,482	145,946	3,830,644	3,682,302	7,367,000
C60/PH60/B14	57,588	57,474	209,821	208,849	174,846	5,463,484	5,285,162	10,573,800

Table 5: Network statistics for the refined bunker algorithm.

We first note from Table 5 that bunker nodes constitute over 99.7% of the total nodes in the networks while the corresponding number for the arcs is 99.5%. Next, we see that preprocessing has allowed a network node reduction of 66-70% compared to the potential network size stated in Section 4.4. Similarly, preprocessing has removed 96-97% of the arcs. When comparing with the extended cargo-bunker network illustrated in Figure 2, we see that the potential node count of the aggregated networks that we use, is 93-97% lower than that of the corresponding extended networks while the arc count is 47-48% lower.

The key values for the standard sequential approach on the 75 instances are reported in Table 6. They are almost the same as for the refined bunker algorithm but the CPU time for solving subproblems now corresponds to the column generation phase of finding routes and schedules without optimizing bunker simultaneously. Likewise, the number of generated columns and the number of calls to the subproblems are derived from the pure routing and scheduling phase. Finally,

we do not report any values on price sensitivity as the bunker prices are not considered while constructing the routes and schedules when using the sequential approach.

	Gap	CPU_{Total}	CPU_{L17}	CPU_{Sub}	Cols	Subs	Filled
C30/PH30/B14	-	7.9	7.8	0.0	88	6	69.7
C40/PH30/B14	-	10.4	10.3	0.0	94	5	76.0
C50/PH30/B14	0.25	13.7	13.5	0.0	121	6	69.5
C50/PH60/B19	0.22	30.3	30.1	0.0	196	10	65.6
C60/PH60/B19	0.12	44.4	44.1	0.0	219	9	65.2

Table 6: Key values for sequential algorithm.

For the sequential approach, 24 out of the 75 instances resulted in a fractional solution, but as can be seen in Table 6 the integrality gap is relatively small. Furthermore, we see that bunker optimization, i.e. rerunning with bunkering included and $L = 17$, is accountable for almost all the CPU time.

Finally, in Table 7 we compare the two approaches to see the effect of integrating bunker. Each entry in the table corresponds to the average over the 15 problem instances generated for the given problem category. The objective function value for the sequential approach serves as a base at which we compare the objective value from the integrated approach. Therefore, the objective values for the sequential approach (Obj) are not reported, and the objective values for the integrated approach (Obj%) are given as the percentage increase from the corresponding sequential objective function values. We do not report the actual objective function values since these are to some extent artificial due to the inclusion of bonus bunker. Both algorithms, however, include this and therefore we can still compare their objective function values. For both algorithms we report the CPU seconds for running the entire algorithm (CPU), the number of cargoes carried in the final solution (Cargoes), and the number of bunker stops in the final solution (Bunker). In the lower part of the table we report the average (Av. Cargo Difference) and maximum (Max Cargo Difference) cargo difference when comparing the solutions found by the integrated approach with those of the sequential approach.

		PH30/B14			PH60/B19	
		C30	C40	C50	C50	C60
Sequential Approach	Obj	-	-	-	-	-
	CPU	7.9	10.4	13.7	30.3	44.4
	Cargoes	16.8	18.4	20.8	28.8	30.4
	Bunker	13.2	14.7	14.7	21.3	21.5
Integrated Approach	Obj%	0.4	0.3	0.7	0.5	0.6
	CPU	17.1	27.3	44.8	123.9	167.4
	Cargoes	16.5	18.3	20.9	29.0	30.0
	Bunker	13.3	14.7	16.1	21.3	21.6
Av. Cargo Difference		0.8	0.9	2.7	3.7	3.7
Max Cargo Difference		2.0	3.0	5.0	6.0	8.0

Table 7: Comparing the two planning approaches.

When comparing the two planning approaches, we see that the percentage increase in profit is relatively small. It is however, important to remember that the fixed costs have not been subtracted, and, hence, we are actually comparing the marginal contributions rather than the profits. The actual profits will therefore be much lower and any difference in profits will correspond to a larger percentage. We, however, do not have data on fixed costs and so, we use the marginal contributions as above. We also note that the profits obtained from the sequential approach are expected to be an optimistic estimate of the standard sequential approach where current practice is to use manual planning in both phases. Furthermore, we note that we are dealing with an industry where numbers are huge. This means that even small percentage increases can lead to

huge increases in profit. Finally, we note that including more ports can help increase the bunker effect as distances between ports will become smaller. As already mentioned, with our setup the distances between ports, and in turn between cargoes, are often so large, that for a given ship, only very few cargoes are actually eligible for transportation.

From Table 7 we also note that the integrated approach does not in general produce solutions that carry more cargoes than the sequential approach. The method is not designed to increase fleet utilization in the sense of carrying extra cargoes. Rather it is designed to increase fleet utilization by carrying the right cargoes and we see that the cargo difference can be as high as 8 cargoes. Aside from the reported cargo differences, we very often found that cargoes carried in both the sequential solution and the integrated solution where carried by different ships in the two solutions.

Overall, we note that a small profit increase can be obtained at little computation time by integrating bunkering in the routing and scheduling planning phase. It should also be noted that such an integration will prevent the construction of bunker infeasible schedules as we saw for many of the larger instances.

8 Concluding Remarks

In this paper we have considered the tramp ship routing and scheduling problem with simultaneous bunker optimization. We have presented a mixed integer programming formulation that extends the standard tramp formulation by accounting for bunkering time, variations in bunker prices and bunker port costs. We have also extended standard formulations by using load dependent cost, speed and bunker consumption. We have developed a solution method that utilizes column generation with a dynamic programming algorithm to generate columns. The devised method is heuristic and this is mainly due to the discretization of the continuous bunker purchase variables. A natural extension of this work is, hence, to solve the continuous version of the problem and also to embed the column generation scheme in a Branch & Bound framework. The method has been devised for a tramp operator that sails full ship loads but the method can be extended to multiple cargoes by changing the subproblem network constructions and the corresponding labels and resource extension functions. In such a case, the load dependent cost, speed and bunker consumption functions would also have to be refined as each ship can then have other load levels than ballast and laden. Generally, the method is very flexible and can be extended to incorporate various operator specific characteristics such as e.g. multiple product types, tank cleaning and restrictions on product successions, by simply changing the subproblem networks and the corresponding labels and resource extension functions.

We have compared the method with a standard sequential approach where routes and schedules are planned without considerations for bunkering. Computational results on 75 generated test instances show that the integrated approach can increase profits slightly. They also show that the decision of which cargoes to carry and on which ships is affected by the bunker integration and by changes in the bunker prices. Consequently, we recommend combining the decisions on fleet scheduling and bunker optimization rather than separating the two planning problems as is current practice.

We also want to mention that the work presented here assumes only one type of bunker even though several exist in practice. It would therefore be very interesting to extent this work to consider multiple types of bunker. Finally, we have solved the problem using a forward curve for the bunker price at each bunker port. If in fact several price scenarios exist, it would be interesting to apply stochastic programming to cope with this price uncertainty.

Acknowledgements

The research presented in this paper has been partly funded by The Danish Maritime Fund and we gratefully acknowledge their financial support. The authors also wish to thank the experienced staff at Maersk Tankers A/S for fruitful discussions and for helping us gain insight into the tramp shipping industry. In particular we would like to thank Jakob Tørring for helpful advice, constructive critique and general support during the project.

References

K. Abdelghany, A. Abdelghany, and S. Raina. A model for the airlines' fuel management strategies. *Journal of Air Transport Management*, 11(4):199–206, 2005.

H. Andersson, M. Christiansen, and K. Fagerholt. The maritime pickup and delivery problem with time windows and split loads. *INFOR*, 49(2):79–91, 2011a.

H. Andersson, J. M. Duesund, and K. Fagerholt. Ship routing and scheduling with cargo coupling and synchronization constraints. *Computers and Industrial Engineering*, 61(4):1107–1116, 2011b.

L. H. Appelgren. A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3:53–68, 1969.

O. Besbes and S. Savin. Going bunkers: The joint route selection and refueling problem. *Manufacturing and Service Operations Management*, 11(4):694–711, 2009.

G. Brønmo, M. Christiansen, and B. Nygreen. Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society*, 58(9):1167–1177, 2007.

G. Brønmo, B. Nygreen, and J. Lysgaard. Column generation approaches to ship scheduling with flexible cargo sizes. *European Journal of Operational Research*, 200(1):139–150, 2010.

M. Christiansen and B. Nygreen. Robust inventory ship routing by column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 7, pages 197–224. Springer, New York, 2005.

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

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Barnhart and L. G., editors, *Transport. Handbooks in Operations Research and Management Science, vol. 14*, chapter 4, pages 189–284. Elsevier, North-Holland, Amsterdam, 2007.

D. W. Darnell and C. Loflin. National airlines fuel management and allocation model. *Interfaces*, 7(2):1–16, 1977.

G. Desauliniers, J. Derosiers, and M. M. Solomon, editors. *Column Generation*. Springer, New York, 2005.

G. Desaulniers, J. Desrosiers, I. Ioachim, M. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57 – 94. Kluwer Academic Publishers, 1998.

G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. Vrp with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 9, pages 225–242. Society for Industrial and Applied Mathematics, 2002.

J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing. Handbooks in Operations Research and Management Science, vol. 8*, chapter 2, pages 35–139. North-Holland, Amsterdam, 1995.

K. Fagerholt and H. Lindstad. Turborouter: An interactive optimisation-based decision support system for ship routing and scheduling. *Maritime Economics and Logistics*, 9(3):214–233, 2007.

K. Fagerholt, G. Laporte, and I. Norstad. Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, 61(3):523–529, 2010.

K. Fagerholt, L. M. Hvattum, T. A. V. Johnsen, and J. E. Korsvik. Routing and scheduling in project shipping. *Annals of Operations Research*, pages 1–15, 2011.

R. A. Gatica and P. A. Miranda. Special issue on latin-american research: A time based discretization approach for ship routing and scheduling with variable speed. *Networks and Spatial Economics*, 11(3):465–485, 2011.

F. Hennig, B. Nygreen, and M. E. Lübbecke. Nested column generation applied to the crude oil tanker routing and scheduling problem with split pickup and split delivery. *Naval Research Logistics*, 59(3-4):298–310, 2012.

S.-H. Hong Lin, N. Gertsch, and J. R. Russell. A linear-time algorithm for finding optimal vehicle refueling policies. *Operations Research Letters*, 35(3):290–296, 2007.

I. Ioachim, S. Gélinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3):193–204, 1998.

S. Irnich. Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. ISSN 01716468.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 2, pages 33 – 66. Springer, 2005.

K. Kobayashi and M. Kubo. Optimization of oil tanker schedules by decomposition, column generation, and time-space network techniques. *Japan Journal of Industrial and Applied Mathematics*, 27(1):161–173, 2010.

J. E. Korsvik and K. Fagerholt. A tabu search heuristic for ship routing and scheduling with flexible cargo quantities. *Journal of Heuristics*, 16(2):117–137, 2010.

J. E. Korsvik, K. Fagerholt, and G. Laporte. A tabu search heuristic for ship routing and scheduling. *Journal of the Operational Research Society*, 61(4):594–603, 2010.

J. E. Korsvik, K. Fagerholt, and G. Laporte. A large neighbourhood search heuristic for ship routing and scheduling with split loads. *Computers and Operations Research*, 38(2):474–483, 2011.

D.-Y. Lin and H.-Y. Liu. Combined ship allocation, routing and freight assignment in tramp shipping. *Transportation Research Part E: Logistics and Transportation Review*, 47(4):414–431, 2011.

S.-H. Lin. Finding optimal refueling policies in transportation networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5034 LNCS:280–291, 2008.

F. Malliappi, J. A. Bennell, and C. N. Potts. A variable neighborhood search heuristic for tramp ship scheduling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6971 LNCS:273–285, 2011.

I. Norstad, K. Fagerholt, and G. Laporte. Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, 19(5):853–865, 2011.

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

H. Oh and I. Karimi. Operation planning of multiparcel tankers under fuel price uncertainty. *Industrial and Engineering Chemistry Research*, 49(13):6104–6114, 2010.

D. Ronen. Cargo ships routing and scheduling: Survey of models and problems. *European Journal of Operational Research*, 12(2):119–126, 1983.

D. Ronen. Ship scheduling: The last decade. *European Journal of Operational Research*, 71(3): 325–333, 1993.

D. Ronen. Marine inventory routing: Shipments planning. *Journal of the Operational Research Society*, 53(1):108–114, 2002.

J. Schönberger, H. Kopfer, and D. C. Mattfeld. A combined approach to solve the pickup and delivery selection problem. In U. Leopold-Wildburger, F. Rendl, and G. Wäscher, editors, *Operations Research Proceedings 2002*, pages 150–155. Springer, 2003.

M. Stålhane, H. Andersson, M. Christiansen, J.-F. Cordeau, and G. Desaulniers. A branch-price-and-cut method for a ship routing and scheduling problem with split loads. *Computers and Operations Research*, 39(12):3361–3375, 2012.

J. S. Stroup and R. D. Wollmer. Fuel management model for the airline industry. *Operations Research*, 40(2):229–237, 1992.

Y. Suzuki. A generic model of motor-carrier fuel optimization. *Naval Research Logistics*, 55(8): 737–746, 2008.

Y. Suzuki and J. Dai. Reducing the fuel cost of motor carriers by using optimal routing and refueling policies. *Transportation Journal*, 51(2):145–163, 2012.

UNCTAD. Review of maritime transport 2011. `http://unctad.org/en/Pages/PublicationArchive.aspx?publicationid=1734`, November 2011.

Z. Yao, S. H. Ng, and L. H. Lee. A study on bunker fuel management for the shipping liner services. *Computers and Operations Research*, 39(5):1160–1172, 2012.

P. P. Zouein, W. R. Abillama, and E. Tohme. A multiple period capacitated inventory model for airline fuel management: a case study. *Journal of the Operational Research Society*, 53:379–386, 2002.