Nested Column Generation Applied to the Crude Oil Tanker Routing and Scheduling Problem with Split Pickup and Split Delivery

Frank Hennig,[1] Bjørn Nygreen,[1] Marco E. Lübbecke[2]

[1] *Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway*

[2] *Chair of Operations Research, RWTH Aachen University, D-52072 Aachen, Germany*

Received 25 January 2010; revised 11 March 2012; accepted 12 March 2012
DOI 10.1002/nav.21489
Published online 13 April 2012 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: The split pickup split delivery crude oil tanker routing and scheduling problem is a difficult combinatorial optimization problem, both theoretically and practically. However, because of the large expenses in crude oil shipping it is attractive to make use of optimization that exploits as many degrees of freedom as possible to save transportation cost. We propose a nested column generation algorithm for this particular split pickup split delivery problem which bears several complexities such as a heterogeneous fleet, multiple commodities, many-to-many relations for pickup and delivery of each commodity, sequence dependent vehicle capacities, and cargo quantity dependent pickup and delivery times. Our approach builds on a branch-and-price algorithm in which the column generation subproblems are solved by branch-and-price themselves. We describe our implementation in the branch-cut-and-price framework SCIP and give computational results for realistic test instances. The high quality schedules we obtain for these instances improve on those in previous studies. © 2012 Wiley Periodicals, Inc. Naval Research Logistics 59: 298–310, 2012

Keywords: maritime transportation; crude oil; split pickup; split delivery; routing and scheduling; branch-and-price; nested column generation

1. INTRODUCTION

In this study, we propose an exact solution approach to a real-world maritime transportation problem called the "crude oil tanker routing and scheduling problem." This is a particularly rich split pickup split delivery problem with time windows and capacity constraints and has been introduced in [10].

Maritime crude oil transportation is a major international transportation mode. In 2010 the crude oil tanker tonnage represented 34% of the total world fleet capacity. The total volume of oil shipped amounted to 2.75 billion tons [17]. A typical vessel engaged in long range, large volume crude oil transportation is the very large crude oil carrier (VLCC). These ships have a length of about 330 meters, a breadth of 60 meters, and can transport 300,000 tons of crude oil at once. Each day by which the duration of a given transportation task may be shortened would save about 50,000 USD at today's fuel price of around 650 USD/ton. A port visit avoided through good routing and scheduling could reduce total cost by 100,000 USD [15]. Through more efficient transportation fleet utilization increases, more cargo can be transported and less harmful emissions per ton mile have to be accepted.

In the type of vehicle routing problem (VRP) considered, vehicles are ships and locations to be serviced are ports. In contrast to the type of VRP, where vehicles operate from a depot and do either pickup or delivery, in maritime crude oil transportation the usually heterogeneous ships start at their present location, pick up and deliver along their route and are not scheduled for a certain destination. Port visits for loading or discharging are time constrained and physical and regulatory circumstances set different limits for ship capacity on different sailing legs. The cargo to be transported is liquid bulk of varying quality or type. Same qualities or types are supplied and demanded in different locations so that a cargo cannot be specified with is origin and destination alone. Also a given transportation task is not necessarily associated with a given shipment volume. Ships may load the quantities they are suited for and that match an efficient routing. This involves the possibility of splitting supplied and demanded quantities in arbitrary ways.

The contribution of our article is twofold. First, we present an accurate model for a logistics problem of major practical importance and develop a solution approach to obtain

Correspondence to: F. Hennig (frank.hennig@iot.ntnu.no) industrially relevant solutions. Second, and probably more important, we propose a general methodology to exactly solve split pickup split delivery problems (SPSDP). The splitting of a customer's demand between several vehicles of a given fleet is a natural attempt to realize the full optimization potential. This is accounted for in the vehicle routing literature by studying the split delivery vehicle routing problem (SDVRP), for which a few exact methods were proposed [3, 5, 6, 9, 12, 13, 18]. The literature on split pickup split delivery problems is even scarcer. We are aware of only two related papers [14, 16], however, both fail to provide a solution approach for our problem. One [14] does not provide us with a state-of-the-art solution approach. The other one [16] covers only splits for explicitly paired pickups and deliveries in a much less complex environment.

Given the limited knowledge on combined split pickup with split delivery, we build on the two recent papers by Desaulniers [6] and Ceselli et al. [5] who propose branch-and-price algorithms for the split delivery vehicle routing problem. These authors exploit one crucial property of the split delivery VRP: There is at most one customer per vehicle route for which only a fraction of the demand is served. Whereas this is no longer true for pickup and delivery problems which permit splitting in both the pickup and delivery locations, we are able to generalize the rationale behind the approach in [5,6].

Our approach is based on integer programming, and we will make heavy use of branch-and-price [2, 8], a methodology we assume the reader to be reasonably familiar with. In particular, we apply a nested column generation approach, a term used in [19] and others to describe a column generation algorithm in which the subproblem is solved by column generation itself. In this spirit, we solve a column generation master problem that comprises all ships in the fleet and solve subproblems, one for each ship, again using column generation. Throughout this article, we will call the restricted all-ship master problem level 1 RMP and the restricted single-ship master problems level 2 RMP. The term subproblem is used to refer to the level 2 RMP route generation subproblem solved by dynamic programming.

The article is structured as follows: In Section 2, we describe the mathematical model thereby introduce some notation. Section 3 motivates and describes the nested column generation concept and problem decomposition. The computational algorithm and its implementation in the SCIP framework is explained in Section 4. Computational results are shown and discussed in Section 5. Finally, we draw conclusions and suggest further research in Section 6.

2. MODEL DESCRIPTION

A heterogeneous fleet $\mathcal{V}$ of vessels or ships has to distribute crude oil between a set of ports. Each port supplies and/or demands one or several types of crude oil, so-called grades $\mathcal{C}$. Each supply/demand has to be picked up or delivered within a certain time window. There may be several distinct time windows to be serviced per port. Denote the set of all time windows of all ports by $\mathcal{N}$. For each ship $v \in \mathcal{V}$, the set $\mathcal{N}_v$ comprises $\mathcal{N}$, and in addition a time window at the initial and at the final location of v, respectively. Formally, the routing of $v \in \mathcal{V}$ takes place on a directed graph $\mathcal{G} = (\mathcal{N}_v, \mathcal{A})$ with arc set $\mathcal{A}$.

Each time window $i \in \mathcal{N}$ calls either for a pickup or a delivery, and is associated with a (supply or demand) quantity Q_i of a particular crude grade $c \in \mathcal{C}$ and a time interval $[\underline{T}_i, \overline{T}_i]$ in which service has to start. We will say that a ship visits a time window, meaning that it visits the associated port within this time window. Supply and demand quantities are specified in weight units and may exceed the ship's capacity. There can be several combinations of pickup time windows to supply a given delivery time window and vice versa. Pickup and delivery ports are concentrated in certain pickup or delivery regions, respectively. Large distances between pickup and delivery regions lead to ship routes having what is called a "voyage" structure: On a voyage a ship may visit one or several pickup regions followed by one or several delivery regions. It is unrealistic that a ship carries cargo from a delivery region back to any pickup region. A ship may undertake several voyages during the planning period, voyage length permitting. The number of time windows a ship services in a region is limited by practical considerations. Therefore the maximum number of different grades simultaneously onboard is limited as well. Because pickup and delivery times may amount to several days for large tanker ships, the quantities picked up and delivered may influence the time feasibility of a route. For time feasibility, it is sufficient that a ship arrives at time window i no later than closing time $\overline{T}_i$. A maximum waiting time between arrival at time window i and the time window's opening time $\underline{T}_i$ may be desired. Sailing times T^S_{ijv} for ship v on arcs $(i, j) \in \mathcal{A}$ are assumed to be deterministic. Ships have a volume capacity based on cargo tank size and a weight capacity due to seaworthiness conditions. The crude grade specific density D_c, $c \in \mathcal{C}$, allows a conversion between weight quantities and volumes. The ship capacity may change during the execution of a route due to external factors like, for example, limited water depth and port regulations. An arc (i, j) may impose a stricter weight capacity limit $\overline{W}_{ijv}$, a stricter volume capacity limit $\overline{V}_{ijv}$, or both, for ship v. Hence, pickup and delivery quantities can be influenced by the order in which different time windows are visited. Set $\mathcal{A}^W_v$ contains all arcs that impose a weight limit on ship v; set $\mathcal{A}^V_v$ contains all arcs with a volume limit for v. Arcs may be in both sets. A ship is allowed to serve any fraction of a time window quantity. Time for service can be calculated based on the per weight unit time consumption T^Q_i. Minimum pickup quantities may apply.

3. NESTED DECOMPOSITION

3.1. Motivation

Recently, Desaulniers [6] and Ceselli et al. [5] very successfully applied branch-and-price algorithms to the SDVRP. They both formulate master problems where each variable corresponds to a combination of a feasible route and a cargo pattern, where the latter represents compatible quantities of demand served for each customer on the route. Building on well-known properties of the SDVRP, they make use of the fact that it is sufficient to consider only cargo patterns where the demand of at most one customer (the split customer) is served fractionally, and all other customers are served either fully or not at all (an analogy holds for the fractional knapsack problem as well). In the language of Dantzig-Wolfe decompostion this means that an extreme point of the polyhedron describing feasible cargo patterns has at most one fractional component. We refer to the quantities associated with these extreme points as extreme cargo patterns. Given a route and a split customer, only two of these patterns are necessary to convex combine all possible fractional quantities in the master problem. This knowledge is instrumental in designing an algorithm for the pricing problem which becomes an elementary shortest path problem (ESPP) where the number of split customers is limited to one. In fact, this adds a helpful "knapsack flavor" to the pricing problem.

The situation is a bit more complicated for the SPSDVRP. The slightly more general statement is still true that in an extreme cargo pattern there is at most one pickup location and at most one delivery location for which the demand is served fractionally. Principally, the label setting algorithm for the ESPP in [6] should not be too difficult to adapt respecting this limitation to two split customers. In contrast, dominance between labels, that is, early elimination of unpromising partial routes is already more complicated. This is because it relies on being able to compare transported quantities en route, and two split locations give more freedom and thus a weaker dominance. In addition, we consider a multicommodity problem, that is, we may have a split pickup and split delivery location for each commodity. As in the ordinary pickup and delivery problem, we may find ourselves in the situation that we cannot decide on feasibility of a route before we see it in total, and this may happen even more frequently with several commodities. This already leads us to not modifying the existing label setting algorithm because it is unlikely that we are able to handle the combinatorial explosion. Even if we had some hope that a combinatorial algorithm would be the right choice, the weight and volume constraints on arcs, particular to the COTRASP, finally bury this. Not only feasibility poses problems, the profits for serving a particular time window can depend on the entire route. There are situations in which it is neither most profitable to go to the volume nor to the weight limit of a particular commodity, see Fig. 1 for an example. Then, reduced costs depend on the amounts loaded and unloaded further on the route, and cannot be computed locally. This, in fact, destroys the "fractional knapsack character" of the pricing problem as exploited in [5, 6]. Needless to say at this point, that loaded and unloaded quantities have an influence on service times and thus on time window feasibility later on the route. These complications together render dominance effectively useless and a label-setting dynamic program is unlikely to be efficient. This is consistent with observations in [4] in a similar situation.

To overcome these difficulties we decided to decouple the routing from the service, that is, we first construct an entire route and then find corresponding quantities to be served at each customer. We retain the key observation by Desaulniers [6] and Ceselli et al. [5] to use extreme cargo patterns, but we need to obtain them with the help of a mixed integer program (MIP). Whereas in the SDVRP a convex combination of only two extreme patterns is sufficient for a given route and a split customer, we may need many more extreme patterns in our SPSDVRP. Candidates are all the extreme points of a polyhedron describing the feasible quantities, and these are far too many to explicitly list. Describing feasible cargo patterns via a polyhedron allows us to formulate even complicated side constraints; but we need to use column generation to generate the extreme cargo patterns for a given route. On the positive side, we can use a straightforward implementation of an ESPP algorithm for routing. In that sense, the decomposition also leads to simplicity and manageability of our approach in the first place, plus it adds a lot more flexibility. As said, on the downside, the overall algorithmic structure gets a little more complicated because we will nest two column generation algorithms. It should be noted that the approach is exact in principle, even though we will use it as a heuristic. The details are given in the following subsection.

3.2. Concept: Three Levels

The algorithm for solving the COTRASP model is based on a decomposition into three levels: On the bottom level we have a route generation subproblem for each single ship. An intermediate level master problem computes (usually several) cargo patterns for each route. Route and cargo patterns together form what we call cargo routes, and the cargo routes from different ships are coordinated on the top level master problem. From top to bottom we number the levels 1 through 3. Although the feasibility of routes, schedules and cargo patterns is ensured on levels 2 and 3, the correct time window quantity split is decided on level 1 via a convex combination of cargo patterns. The restricted master problems at levels 1 and 2 are called *RMP 1* and *RMP 2*, respectively. The following three subsections describe the three levels in more detail. More background on the underlying practical problem can be found in [10].

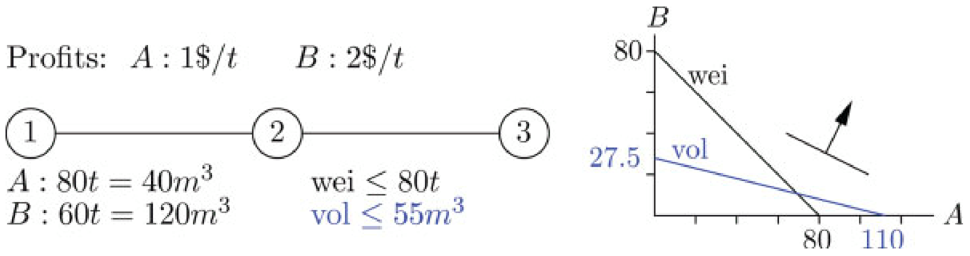

Figure 1. Two products, A and B, are transported from 1 to 3. There are weight and volume constraints on arc (2,3). With a density of $2t/m^3$, product A is much denser than product B with density $0.5t/m^3$. Product B is more profitable regarding weight and one might expect to load as much as possible of product B. An optimal allocation, however, loads only $10t$ of B and $70t$ of A (see figure on right). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

3.3. Level 1 RMP

Each variable (column) of the RMP 1 belongs to a ship v and represents the two parts of a cargo route, the route itself and a route specific cargo pattern. The set of ship v's possible routes is denoted by $\mathcal{R}_v$. $\mathcal{R}_v^* = \mathcal{R}_v \cup \{0\}$ comprises a dummy route 0 which represents that no route is assigned to v. The set of cargo patterns on route $r \in \mathcal{R}_v$ is denoted by $\mathcal{P}_{vr}$. A cargo pattern $k \in \mathcal{P}_{vr}$ consists of feasible quantities Q^P_{ivrk} for pickup or delivery in time window i. Binary variable λ_{vr} equals 1, if ship v sails route r (possibly the dummy route), and 0 otherwise. If ship v sails route r, precisely the (extreme) cargo patterns $k \in \mathcal{P}_{vr}$ are convex combined to yield the actual pickup and delivery quantities on route r. Continuous variables μ_{vrk} represent this convex combination.

The total cost of ship operation includes route cost and the costs for picking up and delivering cargo. The constant C_{vrk} denotes the cost for ship v on route r including service costs caused by cargo pattern k. Let M_i be the minimum number of visits at a time window. M_i can be derived from the total time window amount Q_i and the ship capacities. Let A^N_{ivr} equal 1 if ship v visits time window i on route r, and 0 otherwise. The integer variable z reports the number of nonused ships.

The model can be written as follows:

$$\min \sum_{v\in\mathcal{V}} \sum_{r\in\mathcal{R}_v} \sum_{k\in\mathcal{P}_{vr}} C_{vrk}\mu_{vrk}, \tag{1}$$

$$\sum_{v\in\mathcal{V}} \sum_{r\in\mathcal{R}_v} \sum_{k\in\mathcal{P}_{vr}} Q^P_{ivrk}\mu_{vrk} = Q_i \qquad \forall i \in \mathcal{N}, \tag{2}$$

$$\lambda_{vr} - \sum_{k\in\mathcal{P}_{vr}} \mu_{vrk} = 0 \qquad \forall v \in \mathcal{V}, r \in \mathcal{R}_v, \tag{3}$$

$$\sum_{r\in\mathcal{R}_v^*} \lambda_{vr} = 1 \qquad \forall v \in \mathcal{V}, \tag{4}$$

$$\sum_{v\in\mathcal{V}} \sum_{r\in\mathcal{R}_v} A^N_{ivr}\lambda_{vr} \geq M_i \qquad \forall i \in \mathcal{N}, \tag{5}$$

$$\sum_{v\in\mathcal{V}} \lambda_{v0} - z = 0 \tag{6}$$

$$\mu_{vrk} \geq 0 \qquad \forall v \in \mathcal{V}, r \in \mathcal{R}_v, k \in \mathcal{P}_{vr}, \tag{7}$$

$$\lambda_{vr} \in \{0,1\} \qquad \forall v \in \mathcal{V}, r \in \mathcal{R}_v^*, \tag{8}$$

$$z \in \mathbb{Z}_+. \tag{9}$$

Objective function (1) minimizes the total cost of sailing and service. Constraint (2) ensures that in each time window the sum of all pickups or deliveries made by all ships visiting the time window equals the total time window quantity. The convex combination of (extreme) cargo patterns for each route hereby allows the selection of any pickup or delivery amount feasible on the route. Constraint (3) couples feasible cargo patterns for a particular route and the route itself: Cargo patterns can be convex combined only when their associated route is used. Each ship is allowed to sail at most one route, see constraint (4). Constraint (5) can be used to set a minimum number of visits for each time window. It is used to strengthen the formulation. Constraints (6) and (9) serve the technical purpose to branch on the number z of unused ships (see Section 4.1). Constraints (7) and (8) enforce the correct domains.

Each time a new route is added to the above formulation, a constraint (3) needs to be added as well. This is particularly unfortunate because one can expect the number of routes to grow rapidly. As branching on the binary master variables is not advisable, we substitute for λ_{vr} in constraints (4) and (5) according to constraint (3), relax constraint (8), and drop (3). Constraints (4) and (5) then take the following form:

$$\sum_{r\in\mathcal{R}_v} \sum_{k\in\mathcal{P}_{vr}} \mu_{vrk} + \lambda_{v0} = 1 \qquad \forall v \in \mathcal{V}, \tag{4'}$$

$$\sum_{v\in\mathcal{V}} \sum_{r\in\mathcal{R}_v} \sum_{k\in\mathcal{P}_{vr}} A^N_{ivr}\mu_{vrk} \geq M_i \qquad \forall i \in \mathcal{N}. \tag{5'}$$

Branching is performed on the original problem variables for arcs and nodes of the underlying network. See Section 4.1 for further discussion.

3.4. Level 2 RMP

Each RMP 2, one for each ship v, is a column generator for RMP 1 that provides variables μ_{vrk}. Thus, its objective function is the RMP 1 reduced cost of μ_{vrk} variables. Each RMP 2 accommodates binary routing variables λ_{vr} with the same meaning as in RMP 1. The set $\mathcal{S}_v$ of feasible routes at level 2 is a superset of $\mathcal{R}_v$ because the feasible quantity split is decided only at level 1. Set $\mathcal{S}_v^* = \mathcal{S}_v \cup \{0\}$ again contains a dummy route. Continuous variables q_{iv} represent how much weight is picked up or delivered by ship v in time window i. A μ-column itself is derived from a selected optimal ship route (binary variable λ_{vr}) and optimal pickup/delivery quantities (continuous variables q_{iv}).

Parameter $\delta_{ic} = 1$ signals that time window i supplies or demands grade c. For all other grades $\delta_{ic} = 0$. Parameter I_i indicates the type of time window i, where $I_i = 1$ means a pickup and $I_i = -1$ means a delivery. Continuous variables l_{ijcv} track the weight of grade c onboard ship v on arc (i, j). Parameter A_{ijvr} equals 1 when route r for ship v contains arc (i, j), and 0 otherwise. Continuous variables t_{iv} represent the start-of-service time of ship v at time window i. Only one time variable per time window is needed because we assume that a time window can be visited only once by a particular ship (see Section 3.5).

To compute the level 1 reduced cost at RMP 2, dual information has to be transferred from the first to the second level. Relevant dual information concerns the route and cargo pattern variables λ_{v0} and μ_{vrk}:

$\alpha_i \in \mathbb{R}$	for constraint (2),
$\beta_v \in \mathbb{R}$	for constraint (4′),
$\gamma_i \in \mathbb{R}_+$	for constraint (5′), and
$\eta \in \mathbb{R}$	for constraint (6).

Dual variables β_v and η are constant per ship v. The cost of route r for ship v is denoted by C_{vr}^E, and is computed from arc costs and dual variables γ_i. The cost per weight unit for pickup and delivery in time window i is denoted by C_{iv}^Q which needs to be corrected by the dual variable α_i. An RMP 2 reads

$$\min \sum_{r\in\mathcal{S}_v} C_{vr}^E \lambda_{vr} + \sum_{i\in\mathcal{N}} \left(C_{iv}^Q - \alpha_i\right) q_{iv} - \beta_v - \eta, \quad (10)$$

$$q_{iv} \le Q_i \quad \forall i \in \mathcal{N}, \quad (11)$$

$$\sum_{c\in\mathcal{C}} l_{ijcv} - \overline{W}_{ijv} \sum_{r\in\mathcal{S}_v} A_{ijvr}\lambda_{vr} \le 0 \quad \forall (i,j) \in \mathcal{A}_v^W, \quad (12)$$

$$\sum_{c\in\mathcal{C}} \frac{l_{ijcv}}{D_c} - \overline{V}_{ijv} \sum_{r\in\mathcal{S}_v} A_{ijvr}\lambda_{vr} \le 0 \quad \forall (i,j) \in \mathcal{A}_v^V, \quad (13)$$

$$\sum_{j\in\mathcal{N}} l_{jicv} + I_i\delta_{ic} q_{iv} - \sum_{j\in\mathcal{N}} l_{ijcv} = 0 \quad \forall i \in \mathcal{N}, c \in \mathcal{C}, \quad (14)$$

$$\sum_{r\in\mathcal{S}_v^*} A_{ijvr}\lambda_{vr}\left(t_{iv} + T_i^Q q_{iv} + T_{ijv}^S - t_{jv}\right) \le 0 \quad \forall (i,j) \in \mathcal{A}, \quad (15)$$

$$\underline{T}_i \le t_{iv} \le \overline{T}_i \quad \forall i \in \mathcal{N}_v, \quad (16)$$

$$\sum_{r\in\mathcal{S}_v^*} \lambda_{vr} = 1 \quad , \quad (17)$$

$$l_{ijcv} \ge 0 \quad \forall (i,j) \in \mathcal{A}, c \in \mathcal{C}, \quad (18)$$

$$t_{iv} \ge 0 \quad i \in \mathcal{N}_v, \quad (19)$$

$$q_{iv} \ge 0 \quad \forall i \in \mathcal{N}, \quad (20)$$

$$\lambda_{vr} \in \{0,1\} \quad \forall r \in \mathcal{S}_v^*. \quad (21)$$

Objective function (10) minimizes the total reduced cost. Typically, a single ship services only a subset of time windows and may service only a fraction of a time window quantity. Constraint (11) therefore limits only the maximum pickup and delivery amount in each time window. Constraints (12) and (13) limit the maximum available capacity on used arcs. If an arc is not in use, the load onboard a ship on the arc is kept at zero. Constraint (14) is a flow conservation of loads on arcs. Scheduling constraints (15) and (16) ensure pickup and delivery amounts dependent on port stay times. Constraint (15) allows a ship to start service at arc destination only after arrival at arc destination. Service times in an arc's origin time window are respected in the arc traversal time. Note that (15) is a nonlinear constraint which can be re-formulated in a standard way using a "big M" mechanism: $t_{iv} + T_i^Q q_{iv} + T_{ijv}^S \le t_{jv} + \mathbf{M}(1 - \sum_{r\in\mathcal{S}_v^*} A_{ijvr}\lambda_{vr})$, where $\mathbf{M}$ is the largest possible duration between any possible arrival and any possible departure of ship v at ports associated with i and j. Start-of-service has to take place within the time window specified in constraint (16). Waiting time between arrival at a time window and time window opening cannot be limited by the above model. However, approximate limits can be applied during route generation in the subproblem. At most one route can be selected per ship, as ensured by constraint (17). Variable domain constraints (18) to (21) complete the model.

3.5. Subproblem at Level 3

At the lowest level is the column generation subproblem for the RMP 2 problems. That is, the subproblem furnishes routes, represented by sequences of time windows. It is a rather standard elementary shortest path problem with time windows (ESPPTW). A problem definition and discussion of solution algorithms based on dynamic programming can be found in [7] and in the survey [11]. The objective function is the RMP 2 reduced cost for the λ_{vr} variables that are generated. It comprises real costs for ship operation and port visits, and RMP 2 dual variables associated with constraints (12), (13), (15), and (17).

Due to the large distance between pickup and delivery regions it is not time feasible for a single ship to visit a time window on several occasions on consecutive voyages. Even visiting a time window more than once during a single voyage is considered unrealistic even though it could be beneficial. The so-called resource extension functions of the subproblem extend the routing cost and time, and information about grades onboard. Cost is a nonmonotonic resource whereas time is a strictly increasing resource. Time is extended based on deterministic sailing times and smallest possible service times required in real operations. Because pickup and delivery quantities determine the actual service time, the time window arrival times calculated this way are only approximate. Because routes with extraordinary waiting times are not judged reasonable in shipping, we enforce maximum waiting time limits for the duration between arrival at a port and start of service (even though theoretically, this may discard optimal routes). Again, the time limit can only be approximate. Grade information has to be tracked on partial paths. An extension of a path to a delivery time window is infeasible if a demanded grade is not available in already visited pickup time windows. Similarly, a path cannot be extended from a delivery time window to a pickup time window, that is, to the beginning of the next voyage, if for a grade picked up on the present voyage there is no matching delivery time window on that voyage.

Label dominance is based on cost, time, visited time windows, and grade information. A label L_1 containing all necessary information about a partial path can dominate a label L_2 at the same node, if total cost of L_1 is no larger than the cost of L_2, start of service at the node for L_1 is no later than for L_2, the path of L_1 visits a subset of nodes visited on the path of L_2, and L_1 and L_2 have identical sets for pickup grades and delivery grades. Barring grade compatibility, the described dominance is the same as for the (E)SPPTW.

The main idea of dominance is that all extensions for label L_2 are also feasible for label L_1. In addition L_1 may have more possible extensions. Let us consider a single voyage: for the considered pickup and delivery problem with multiple products the visited pickup time windows in a pickup sequence determine which delivery time windows are allowed to be visited and which sort of delivery time windows have to be visited. If, for example, a pickup time window supplying grade A is visited, then all delivery time windows demanding grade A are eligible for a visit. In fact, at least one delivery time window demanding grade A has to be visited. If for instance L_1 visits a subset of pickup time windows visited by L_2 and can only pick up grades A and B in these time windows instead of A, B, and C available in L_2, extensions of L_1 visiting delivery time windows with the demanded grade C are not allowed. Hence, L_1 cannot be extended in the same way as L_2. If pickup grades are identical, but delivery grades are not, L_1 has to visit a certain sort of delivery time window that L_2 already has visited. Also in this case dominance is not possible.

We omit the rather standard mathematical formulation of the subproblem and the technical description of the dynamic program in this article. In our implementation, we use the resource constrained shortest path algorithm provided by the boost C++ graph libraries (www.boost.org).

4. COMPUTATIONAL ALGORITHM

The nested column generation algorithm has been implemented in C++ using the noncommercial branch-cut-and-price framework SCIP (scip.zib.de) developed by Achterberg [1]. SCIP can be used as a standalone mixed integer programming solver as well and provides a widely extensible state-of-the-art tree management. Column generation is directly supported via SCIP's plugin-based architecture, and nested column generation is no problem either, even though several useful functionalities concerning branch-and-price need to be (and can be) added for the problem at hand. Even though there may be a small overhead, for a proof-of-concept SCIP is a very reasonable choice.

In Section 4.1, we describe our efforts to handle our type of branching in SCIP. After explaining the initial algorithm setup in Section 4.2, we describe the level 1 pricing strategy in Section 4.3. For the latter, a pattern generation heuristic is applied that is sketched in Section 4.4. Finally, Section 4.5 mentions a few actions taken to accelerate the algorithm.

4.1. Branching

It is well known that branching (in either master problem) should not be done on the route variables directly because the down-branch is ineffective and difficult to handle in the subproblems. Moreover, this would lead to an unbalanced branch-and-bound tree. Instead, we branch on fractional flows on arcs and nodes, the so-called "original variables," calculated as follows:

$$x_{ijv} = \sum_{r \in \mathcal{R}_v} A_{ijvr} \sum_{k \in \mathcal{P}_{vr}} \mu_{vrk} \qquad \forall v \in \mathcal{V}, (i,j) \in \mathcal{A}, \tag{22}$$

$$y_{iv} = \sum_{r \in \mathcal{R}_v} A^N_{ivr} \sum_{k \in \mathcal{P}_{vr}} \mu_{vrk} \qquad \forall v \in \mathcal{V}, i \in \mathcal{N}. \tag{23}$$

Parameter A^N_{ivr} equals 1 if ship v on route r visits time window i, and 0 otherwise. We impose integrality on variables $x_{ijv} \in \{0,1\}$ for arcs (i,j) and $y_{iv} \in \{0,1\}$ for nodes i for each ship. We can therefore directly use various variable selection and branching strategies build-in in SCIP. These additional constraints, together with their dual variables, are omitted in

Section 3.4 for reasons of clarity. Note that branching an x_{ijv} or y_{iv} to zero eliminates the corresponding already generated μ_{vrk} variables from the restricted master problem. The inclusion of the additional dual variables ζ_{ijv} for constraint (22) and ξ_{iv} for constraint (23) in the RMP 2 objective function is straight-forward. Branching in the RMP 2 is done analogously. There, the arc and node constraints take the same form as constraints (22) and (23) but λ_{vr} replaces term $\sum_{k\in\mathcal{P}_{vr}} \mu_{vrk}$.

4.2. Algorithm Setup

The initial setup consists of an "empty" RMP 1, one "empty" RMP 2 for each ship and a structure for the dynamic program (DP). Each route set $\mathcal{R}_v^*$ only contains the dummy route index 0. All sets $\mathcal{R}_v$ are empty. To ensure feasibility of the problem initially and during branch-and-price, we introduce one artificial variable for each time window. The coefficients of these continuous variables equal to Q_i for time window i in constraint (2) and M_i for the minimum visit constraint (5). These artificial variables are penalized in the objective function so that no artificial variable appears in an optimal solution. In all RMP 2, sets $\mathcal{S}_v$ and $\mathcal{S}_v^*$, respectively, are initially identical with RMP 1 sets $\mathcal{R}_v$ and $\mathcal{R}_v^*$. The dynamic program is identical in structure for each ship. Because the cost and feasible arc set for the dynamic program have to be updated each time an RMP 2 is solved, we maintain only one dynamic program the parameters of which are adjusted each time it is called.

In the following two sections, we explain the pricing strategy and the actual generation of column for the first level.

4.3. Level 1 Pricing Strategy

The RMP 1 pricing problem is to identify a minimum reduced cost column, that is a least cost extreme cargo pattern as described in Section 3.1, with respect to certain dual variable values. However, an often faster (heuristic) strategy in branch-and-price algorithms is to just identify a number of negative reduced cost columns instead of solving the subproblem to optimality. In our approach, this is even more important because we experienced tail-off effects that can consume large amounts of time when solving the level 2 column generation algorithm. Therefore, it seems natural to terminate the level 2 optimization as soon as we find a few negative reduced cost solutions that are routes with compatible cargo patterns.

The RMP 1 pricing strategy is described in Procedure 1, which we call *SolveNodeLevel1*. It illustrates solving the LP relaxation at a node in the RMP 1 branch-and-bound tree to optimality. The actual pricing for a particular ship is sketched in Procedure 2, called *ExecutePricing*. Procedure 2 initializes a given ship's RMP 2 and solves it by branch-and-price, identifies, if desired, a set of cargo patterns, and extends the level 1 RMP with one or several new cargo patterns. Procedure 1 has as input two global (fixed) parameters, t^{lim} and c^{lim}. For each RMP 2 we set an optimization time limit t^{lim}. The solution of a RMP 2 is stopped as soon as the time limit is reached or an improving column with reduced costs less than negative threshold c^{lim} is found. We reach LP optimality of a RMP 1 node when no improving column for any ship could be identified, and all RMP 2 optimization times are less than t^{lim}.

Procedure 1 *SolveNodeLevel*1$(t^{\text{lim}}, c^{\text{lim}})$

```
1.  $q \leftarrow \emptyset$
2.  **for** each $v \in \mathcal{V}$ **do**
3.      $q.enqueue(v)$
4.  **end for**
5.  $success \leftarrow$ **true**
6.  **while** $success =$ **true do**
7.      $GetBranching(x, y)$
8.      $SolveRelaxation(\{\mathcal{P}_{vr} : v \in \mathcal{V}, r \in \mathcal{R}_v\}, x, y)$
9.      $p \leftarrow q$ // initialize ship queue with last iteration's
        result queue
10.     $t \leftarrow 0$
11.     **while** $p \neq \emptyset$ **do**
12.         $p.dequeue(v)$ // Consider first ship in queue
13.         $T \leftarrow \max\{t^{\text{lim}}, t_v\}$
14.         $(\bar{c}_{\min}, t) \leftarrow ExecutePricing(v, T, c^{\text{lim}})$
15.         **if** $\bar{c}_{\min} < 0$ **then**
16.             $success \leftarrow$ **true**
17.             $q.remove(v)$
18.             $q.enqueue(v)$ // Queue ship in result queue
19.             **break**
20.         **else**
21.             $success \leftarrow$ **false**
22.             **if** $t < t^{\text{lim}}$ **then**
23.                 $q.remove(v)$
24.                 $q.enqueue(v)$ // Queue ship in result queue
25.             **else**
26.                 $p.enqueue(v)$ // Queue ship at the end of
                    current queue
27.                 $t_v \leftarrow \infty$
28.             **end if**
29.         **end if**
30.     **end while**
31. **end while**
```

Procedure 1 is best explained by walking through an example: Given a set $\{A, B, C, D\}$ of ships, Procedure 1, in lines 1 to 1, first defines a processing order for each RMP 2 to be solved. This is done in forming a queue $q = (A, B, C, D)$. At the beginning of each pricing iteration the linear relaxation based on the current sets of columns $\mathcal{P}_{vr}$ for all ships and routes and the current branching situation for variables x_{ijv} and y_{iv} is solved (line 1). Columns representing routes are omitted as mentioned in Section 3.3. Each pricing iteration (line 1) may have several pricing rounds (line 1) depending on how many level 2 RMPs are solved. A pricing round represents the execution of a level 2 RMP. The goal is to stop a pricing iteration as soon as a level 2 RMP delivers at least one improving column (line 1) or if no improving columns can be found for any ship.

The ship queue to be processed in the current pricing iteration is called p and the ship queue for the following iteration is called q. Queue p is initialized with q coming from the previous iteration or the initial definition described above. The size of p at processing start equals the number of ships $|\mathcal{V}|$, which is also the principle limit of pricing rounds of the ship processing. The components t_v of vector t represent ship specific level 2 optimization time limits. These time limits are initialized with 0. In the first pricing round ship $v = A$ is dequeued from p reducing queue p from (A, B, C, D) to (B, C, D). Queue $q = (A, B, C, D)$ remains unchanged. The actual RMP 2 optimization limit T is set to $t^{\lim}$. Then pricing is executed for ship A through procedure *ExecutePricing*. The pricing procedure returns the minimum reduced cost $\overline{c}_{\min}$ found by the second level and the actual level 2 optimization time t.

If $\overline{c}_{\min}$ is negative, new improving columns have been found and the level 1 RMP has been extended by Procedure 2. Ship A is removed from next iteration's queue q and enqueued at the end of the queue: $q = (B, C, D, A)$. The pricing iteration is completed and the whole process starts over again with LP relaxation solving. In the case where no improving columns for A exist, that means $\overline{c}_{\min} = 0$ and $t < t^{\lim}$, ship A is also queued at the end of q but a new pricing round has to be started with the next ship in current queue p, namely B. Let us assume the latter case and *ExecutePricing* is executed for ship B. At this point p equals (C, D) and q still equals (B, C, D, A). Now assume that $\overline{c}_{\min}$ takes on value 0 and the time limit is reached. In this case no conclusion can be drawn and ship B is enqueued in $p = (C, D, B)$. This way B may be examined again but next time without the optimization time limit. In the worst case pricing rounds for C and D lead to no conclusion either. The current queue would then be $p = (B, C, D)$ and B would have to be examined again. Of course, without the optimization limit a clear conclusion can be drawn after each round.

Procedure 2 facilitates the level 2 branch-and-price. Before the level 2 problem can be solved, procedure *FixLevel2-Variables* (line 2) fixes all RMP 2 branching variables x_{ijv} for arcs and y_{iv} for nodes already fixed in the RMP 1 to 0 or 1. Procedure *UpdateLevel2Objective* (line 2) updates the dual information α_i, β_v and η in the RMP 2 objective and changes

Procedure 2 *ExecutePricing*$(v, T, c^{\lim})$

```
 1. $GetBranching(x_v, y_v)$
 2. $GetDuals(\boldsymbol{\alpha}, \beta_v, \boldsymbol{\gamma}, \eta, \boldsymbol{\zeta}_v, \boldsymbol{\xi}_v)$ // Get dual information
 3. $FixLevel2Variables(x_v, y_v)$
 4. $UpdateLevel2Objective(\boldsymbol{\alpha}, \beta_v, \boldsymbol{\gamma}, \eta, \boldsymbol{\zeta}_v, \boldsymbol{\xi}_v)$
 5. $UpdateDPCost(\boldsymbol{\gamma}, \boldsymbol{\zeta}_v, \boldsymbol{\xi}_v)$
 6. $(\overline{c}_{\min}, t, \mathcal{C}) \leftarrow SolveLevel2(c^{\lim}, T)$
 7. if $\overline{c}_{\min} < 0$ then
 8.    for each $r \in \mathcal{C}$ do
 9.       if $r \notin \mathcal{R}_v$ then
10.          $\mathcal{R}_v \leftarrow \mathcal{R}_v \cup \{r\}$
11.          $Create(\mathcal{P}_{vr} \leftarrow PatternGeneration(r))$
12.       else
13.          $\mathcal{P}_{vr} \leftarrow \mathcal{P}_{vr} \cup GetPatterns(r)$
14.       end if
15.       ExtendLevel1RMP
16.    end for
17. end if
    return $\overline{c}_{\min}$
```

the route costs C^E_{vr} for all columns already contained in $\mathcal{S}^*_v$ based on the new dual information ζ_{ijv} on arcs, and ξ_{iv} and γ_i on the nodes. Finally, the same dual information is used by procedure *UpdateDPCost* (line 2) to change the arc costs in the dynamic program. After this initialization step procedure *SolveLevel2* executes the level 2 branch-and-price in line 2. Procedure *SolveLevel2* is interrupted as soon as the first negative RMP 1 reduced cost column better than $c^{\lim}$ is found or when the time limit is reached. At the outset of Procedure *SolveLevel2* a quick test is done to check if any column already known from previous pricing iterations for the considered ship has a reduced cost smaller than $c^{\lim}$. If that is the case, branch-and-price is not executed. The actual optimization time is recorded in t. This t does not have any impact on Procedure 2 and is only transferred to Procedure 1 (see line 1 in Procedure 1). All routes from feasible solutions yielding a negative reduced cost column are recorded in set $\mathcal{C}$. Because *SolveLevel2* is a column generation procedure, route sets $\mathcal{S}_v$ and $\mathcal{S}^*_v$ grow during optimization. In general, both sets grow during each pricing round, and of course all routes found in previous iterations in all predecessor tree nodes are kept and need not be generated again by the subproblem.

The value $c^{\lim}$ is negative only if improving RMP 1 columns have been identified. A level 2 solution returns the route used and a corresponding feasible cargo pattern, that can be used as RMP 1 column. Set $\mathcal{C}$ may include one or several different routes. For a single route, negative reduced cost solutions may exist with one or several different cargo patterns. To potentially reduce the number of RMP 1 pricing iterations we use a heuristic cargo pattern generator, *Pattern-Generation*, to identify, for each route found, several feasible cargo patterns at once. If a route is not yet contained in the RMP 1 route set $\mathcal{R}_v$ we include the route (line 2) and create a new cargo pattern set $\mathcal{P}_{vr}$ filled with a number of feasible cargo patterns (line 2). It may happen that the resulting columns may not all have negative reduced cost but together span a polyhedron that describes feasible cargo patterns for the given route. In the best case this region will be sufficiently large to provide all pickup and delivery quantities needed during level 1 optimization. Should the region be too small, $\mathcal{C}$ will at a future iteration, comprise a route already contained in $\mathcal{R}_v$. To avoid an accidental generation of identical columns we do not execute *PatternGeneration* in this case and only include all additional cargo patterns that yielded negative reduced cost solutions (line 2). Finally, Procedure 2 extends the level 1 RMP and returns the minimum reduced cost identified.

The complete level 2 branch-and-price is solved in SCIP with default settings. The dynamic program is called in each tree node as long as negative reduced cost columns can be identified. All columns with negative reduced costs provided by the dynamic program in one pricing iteration are included in sets $\mathcal{S}_v$ and $\mathcal{S}^*_v$.

4.4. Cargo Pattern Generation

Given a route, it is simple to find a number of different cargo patterns heuristically. These are not necessarily extreme but hopefully provide a reasonable approximation. If needed, extreme cargo patterns are generated if the heuristic fails. We apply a strategy that lends itself from the less complicated subproblem described in [6]. We assume that all routes contain at most two voyages, which is consistent with the realistic data available. The procedure works as follows:

1. Find all grade feasible pairs of one pickup and one delivery time window for each voyage in the known route.
2. Define all sequences of pairs for each voyage, where a sequence is a unique ordering of pairs.
3. For each sequence greedily service as much cargo as possible for the first, then second, then third, etc. pair in the sequence.
4. Combine all non-identical cargo patterns found that way for the voyages in all possible ways.

This procedure ignores the fact that it may not be optimal to service as much cargo as possible for a given pair as explained in Section 3.1. It also ignores that for different times of beginning the second voyage, different cargo patterns may be obtained. This enables us to generate cargo patterns for each voyage independently and combine these patterns afterwards. An empty cargo pattern is also assigned to each route, which represents pickup and delivery quantities equal to 0 in each time window.

The cargo pattern generator takes into account all constraints included in level 2. Once a cargo quantity for a pair is decided, time window feasibility along the route must be ensured. Otherwise a cargo pattern would not be feasible. To be able to do this test for the second voyage, we assume maximum possible pickup and delivery times on the first voyage and thus can derive a latest start of voyage time for the second voyage.

The number of cargo patterns generated this way may be large. A small example may clarify this. A single voyage with only two pickup and two delivery time windows supplying and demanding the same grade has four time window origin-destination pairs. Hence there are $4! = 24$ possible sequences with at most 24 different cargo patterns. If a route contains two such voyages already 24 times 24 cargo patterns are possible. Clearly, many of the single voyage cargo patterns may be redundant, but especially for longer routes with few grades the number of unique cargo patterns grows substantially. To avoid unreasonably large matrices of the linear programs to be solved in each node of the branching tree, we allow to specify a maximum number of cargo patterns to be included into the RMP 1 in one pricing iteration.

4.5. Acceleration

SCIP provides a number of heuristics to identify feasible solutions used at different stages of the algorithm. In addition to these default heuristics we include two simple heuristics, one at each level: On level 1 we consider the LP relaxation solution at each pricing iteration and try to find a feasible solution in the following way: Identify which (fractionally) used route of which ship is used the most. Fix all columns of that ship not corresponding to that route to 0 and reoptimize the LP problem. Continue this until one route for each ship has been selected and check if the resulting solution is feasible with respect to the total quantity to transport. On level 2 we know that each route in route set $\mathcal{S}$ and each route provided by the dynamic program yields a feasible solution in the RMP 2. However, these solutions may not be identified during branch-and-price. Therefore we apply a fast probing procedure provided by SCIP to solve the LP relaxation with each route in $\mathcal{S}$ at the beginning of Procedure 2 and each route provided by the dynamic program one at a time fixed to 1.

A further acceleration can be achieved by solving the dynamic program with a heuristic dominance rule. Whenever the heuristic dominance rule does not yield negative reduced cost columns, the correct dominance rule can be applied. The heuristic dominance rule we apply only takes into account costs and timing when comparing two partial paths. The information about grades picked up and delivered is ignored.

5. COMPUTATIONAL RESULTS

5.1. Instances and Settings

We use the same reality related test instances as in Hennig et al. (working paper). The test instances are grouped into two sets. The smaller instances described in Table 1 are of varying complexity. The time periods of the first three instances only allow routes with single voyages. In contrast, instances 4 to 6 require routes with up to two voyages. Table 2 illustrates the larger instances. Each instance, starting from instance 8, has a few additional time windows and ships as compared to its previous instance.

We run SCIP 1.1.0.6 with CPLEX 11.2 as LP solver on 64-bit PCs with Intel Core 2 Duo CPUs, 3 GHz, 6 gigabyte RAM (only one core is used per instance) and linux operating system. Two instances are run simultaneously on the same computer as long as there is sufficient memory. We let the branch-and-price run for 12 hours. Because of the timing of the running time check points, longer branch-and-price times are possible. However, this does not have any significant impact on the solution. During branch-and-price we have to disable SCIP's preprocessing, cut generation, and propagation functionalities to ensure a correct dual bound. For $t^{\lim}$ we choose 60 seconds and $c^{\lim} = -1$. We allow at most 30 cargo patterns per voyage to be selected in the pattern generator. After 12 hours we solve the resulting level 1 master problem without pricing with default CPLEX. In this second run we solve the original problem comprising objective function (1) and constraints (2) to (9) with all routes and cargo patterns found during the first run. Preprocessing and cut generation can be applied. Branching is performed on variables λ_{vr} and z.

5.2. Results and Discussion

The computational results are shown in Tables 3 and 4. We assess the results with respect to solution quality, algorithm performance, and capability of the underlying model. In the result tables, we can distinguish between results based on the first branch-and-price run carried out with SCIP and the second run without pricing performed with CPLEX. The first part of each table, above the broken line, is dedicated to SCIP. The parts under the broken lines report the CPLEX information.

Table 1. Instance set A.

Instances	1	2	3	4	5	6
No. of ships	2	3	5	2	4	6
No. of products	2	2	5	4	5	5
No. of loading time windows	2	3	6	5	6	6
No. of discharging time windows	5	8	14	8	12	16

Table 2. Instance set B.

Instances	7	8	9	10	11	12
No. ships	8	8	9	10	12	15
No. products	4	5	6	7	8	11
No. loading time windows	9	12	14	15	18	21
No. discharging time windows	9	13	15	18	19	26

Before we start the discussion, the following explanations may help to read the result tables: For the first run the best lower bound (at optimization time limit) and the objective value at root node LP relaxation optimum are reported. The root node gap is the relative gap between the best solution and the root node LP relaxation. This gap gives a measure for the tightness of the LP relaxation. In fact, best solution refers to the best solution identified by CPLEX. Therefore we do not report optimality gaps between the best lower bounds and the best solutions, because best solutions have in most cases not been identified in the first run. Pricer calls take place whenever new dual information is available during the solving of a node in the branching trees. The number of calls is reported for the first level RMP and for all second level RMPs in total. Route regenerations refer to the number of times level 2 provides a route that is already contained in the first level RMP. The time needed to find all columns contained in the best solution is reported under time to best cargo pattern set. Under the broken lines in the tables the number of explored nodes and the total running time for the second run are reported. Except for instances 1, 2, and 4 the time to best solution is the sum of the time used for branch-and-price and the time until CPLEX finds the best solution.

Let us first take a look at the reported (best) solutions: In all instances except instance 6 and 12 the entire available fleet has been used. Optimality is only proven for instances 1, 2, and 4. All other instances may not be optimal. Optimality gaps are not reported for two reasons. First, for instances 8 to 12 RMP 1 root node LP relaxations have not been solved to optimality in the given time limit and thus no valid lower bounds are provided. Second, only instances 1, 2 and 4 provide interesting feasible solutions during the first run. For all other instances valuable solutions are first identified in the second run. However, where applicable we provide root node gaps that are essentially identical to the optimality gaps. The average gap for the multiple voyage instances 4 to 7 is only 4.9%, which is quite small for the freedom provided by the model. It is conspicuous that the relaxation for instances 1 to 3 allowing only one voyage per route is much tighter than for the instances with longer time periods. However, a general conclusion should not be drawn. The quality of the results greatly depends on the routes and cargo patterns generated for the first level RMP. Hence, the performance of the algorithm especially on the second and subproblem level plays an important role.

Table 3. Instance set A.

Instances	1	2	3	4	5	6
No. of ships used in best solution	2	3	5	2	4	4
Best solution	2681	4112	6936	5121	8102	7587
Best lower bound	2681	4112	6831	5121	7702	7416
Root node LP relaxation	2670	4088	6831	4804	7702	7408
% root node gap	0.4	0.6	1.5	6.6	5.2	2.4
No. of RMP 1 pricer calls	48	439	802	913	586	254
No. of RMP 2 pricer calls	614	19,591	13,239	33,633	26,975	28,711
No. of RMP 1 cargo patterns	540	12,698	29,122	54,399	44,745	21,469
No. of RMP 1 routes	44	449	1124	654	640	345
No. of route regenerations	0	0	60	89	58	15
Time to best cargo pattern set (s)	1	143	30,733	2499	685	38,177
Total branch-and-price time (s)	2	496	43,260	4262	43,200	43,380
% of B&P time used by level 2	100	96	100	96	100	100
% of RMP 2 time used by DP	4.7	3.0	3.2	2.7	1.8	3.5
No. of explored nodes on level 1	22	161	2	392	35	11
No. of explored nodes on level 2	506	17,325	8499	31,493	34,869	22,335
No. of explored nodes in final branching	*N/A*	*N/A*	232,583	*N/A*	31,925	435
Additional branching time (s)	*N/A*	*N/A*	10,036	*N/A*	2201	19
Time to best solution (s)	2	156	46,915	3550	45,390	43,398

Performance can be judged by the number of routes and patterns generated and the time this takes. It turns out that the number of RMP 1 pricer calls clearly is limited by the time needed on level 2. Basically, the entire optimization time is consumed by the second level. Pricing iterations there are significantly faster. Compared to level 1, many pricer calls are executed, that means many calls to the dynamic program are made. However, at most 5.8% of the level 2 optimization time is consumed by the dynamic program. The sum of the RMP 1 routes and of the route regenerations for all instances exceeds the number of RMP 1 pricer calls. That means, that with each RMP 1 pricing iteration at least one route with most often many cargo patterns is included into the RMP 1. A route is included several times if it is beneficial to extend the existing polyhedron of cargo patterns for that route. The numbers show clearly that there are many unique cargo patterns for the identified routes. Cargo pattern generation, which is part of the first level, consumes negligible time. With at most 30 cargo patterns per single voyage or 900 patterns per two-voyage-route we allow the RMP 1 to grow fast with each new route. Still route regeneration takes place especially for the larger instances and shows that cargo

Table 4. Instance set B.

Instances	7	8	9	10	11	12
No. of ships used in best solution	8	8	9	10	12	14
Best solution	8471	10,405	11,471	12,927	14,523	16,998
Best lower bound	8056	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
Root node LP relaxation	8041	—	—	—	—	—
% root node gap	5.2	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
No. of RMP 1 pricer calls	975	259	345	259	294	360
No. of RMP 2 pricer calls	71,497	20,709	13,788	4135	1686	2302
No. of RMP 1 cargo patterns	12,387	6893	9248	7647	9155	7921
No. of RMP 1 routes	888	289	340	299	309	380
No. of route regenerations	148	69	99	55	79	87
Time to best cargo pattern set (s)	7812	38,124	39,914	15,082	24,398	35,148
Total branch-and-price time (s)	43,200	43,260	43,320	43,200	47,100	43,260
% of B&P time used by level 2	100	100	100	100	100	99.6
% of RMP 2 time used by DP	1.3	2.5	2.2	5.8	3.6	1.2
No. of explored nodes on level 1	147	1	1	1	1	1
No. of explored nodes on level 2	59,540	15,528	10,323	1807	761	962
No. of explored nodes in final branching	817,400	9834	78,290	30,449	2,317,267	1,028,948
Additional branching time (s)	5408	73	797	305	17,237	10,968
Time to best solution (s)	3835	43,320	44,072	43,485	50,539	53,385

patterns are not easily obtained. The fractions of total time used by level 1 underline that producing many patterns poses no problem with LP relaxation solving.

The tree sizes of the second level reflect the complexity of nested column generation. We try to reduce time consuming tail-off problems by stopping optimization soon after the first improving solutions have been found. Still we face another time problem. If we consider the number of times level 2 pricing has been executed and the number of processed nodes on level 2 in total, we see that on average there are only few pricer calls in each tree node. Subproblem solving consumes comparable little time and so time consumption is mainly caused by solving level 2 LP relaxations. We observe a few cases in each of the largest instances where solving a single RMP 2 takes more than 1000 seconds. If we could disregard these cases as problematic exceptions we still would have average single node LP solving times for the largest instances of up to 42 seconds.

The second run underlines the complexity of the largest instances. Although all columns are known, and presolving and cut generation are applied, CPLEX needs ∼3 to 4 hours to solve instances 11 and 12. In fact, instances 8 and 12 running alone on a single computer could not be solved to optimality due to insufficient memory. The optimizations terminate with 1.6% and 3.2% optimality gap.

There are two key challenges to the proposed algorithm: time consumption on level 2 and the ability to find good solutions during the first run. A technical way to improve running times would be to solve all RMP 2 for one pricing iteration in parallel. This would eliminate the need for parameter t_{lim} and a large speed up may be obtained because often one of the RMP 2 solves relatively quickly, while others may need exceptional running times. A more elaborate pattern generation may also reduce running time through reducing the number of route regenerations. At least it is not advisable to replace the proposed pattern generation strategy with solving an LP for a given route for different sets of dual variable values or omit the pattern generation completely. Both approaches are significantly slower than the proposed one. It may also be beneficial to reduce the RMP 1 and RMP 2 matrices through omitting arc and node constraints and implicitly branching on arcs and nodes through branching on sets of columns. This however, would have prevented us from using advanced branching techniques and heuristics already implemented in SCIP. Good solutions have to be found heuristically. The employed RMP 1 heuristic mentioned in Section 4.5 helps to find solutions but do not work well enough to replace the second run. A few instances show a considerably shorter time to best cargo pattern set than to algorithm termination as is the case in many branch-and-bound contexts. This should be exploited.

The underlying model provides great freedom. We have limited the routing possibilities only with respect to realistic constraints and not to achieve a faster algorithm. If a larger number of potentially visited time windows on a voyage is desired, the algorithm would remain the same. Arbitrary splitting of time window quantities is permitted. In (Hennig et al., working paper) the same test instances were run with two column pregeneration algorithms, one allowing arbitrary splitting of cargo quantities and the other one based on discretized splits. With the first algorithm no feasible solution was found for many of the larger test instances. For all but one of the other instances we now improve results slightly and in one case up to 8%. We improved the results compared to the second algorithm in (Hennig et al., working paper) between 1 and 7%. However, we did not find better solutions for the two largest instances. In both instances we now use one additional ship which leads to a cost increase of 5% for instance 12. Interestingly, costs for instance 11 do not increase due to one voyage less in the new solution. Note that the costs reported in Tables 3 and 4 are not directly comparable to the results reported in Hennig et al. (working paper). The above discussion is based on a postoptimization step similar to the one described in Hennig et al. (working paper). It has to be pointed out that the proposed branch-and-price approach considers a significantly larger feasible region than the other mentioned models. The same feasible region would not have been possible to consider in a column pregeneration context.

6. CONCLUSIONS AND FURTHER RESEARCH

In this article, we present a rich routing and scheduling problem eminent in today's industry. Unlike classical assumptions in the vehicle routing literature, we consider a heterogenous vehicle/ship fleet, multiple commodities, and time window service quantity dependent service times. The problem includes many-to-many relations for pickup and delivery of single commodities and time window sequence dependent vehicle capacities. We approach this problem with a nested column generation algorithm decomposing the problem into many single ship problems which are themselves decomposed into a route generation and a cargo decision problem. This way we avoid a dynamic program, that is possibly extremely difficult to solve. Our results demonstrate the capabilities of the approach.

For the SDVRP, the largest instances optimally solvable today comprise about 100 customers [6]. Allowing a split of loads in both the pickup and delivery time windows implies a complication of this already tremendously difficult optimization problem. The problem gets even more complex with arc dependent capacity constraints and quantity dependent time window service times. As a computational consequence this implies a further combinatorial explosion of solution possibilities, so it is not surprising that the resulting type of SPSDP is even harder to solve. It is precisely this explosion (in finding optimal combinations of split loads) which consumes most of our reported computation times. Thus, being able to provide high quality solutions to instances with almost 50 pickup and delivery locations (as we do) is noteworthy.

Our research immediately points to further research directions. Even though we have a satisfactory proof-of-concept, there is still room for improvement. We did not, for instance, add valid inequalities to any restricted master problem in order to strengthen the dual bounds, nor did we implement tailored primal heuristics to be used in our nested decomposition approach. More elaborate branching rules for the SPSDP could help solving the RMP 2 level quicker. It would be interesting to study the SPSDP from a generic point of view. A further very interesting subject of study should be to find efficient algorithms to solve the cargo pattern generation problem quickly and in an exact fashion. Efficient branching rules for the SPSDP are also to be developed.

ACKNOWLEDGMENTS

Most of this work was done while the first and third authors were with the Combinatorial Optimization & Graph Algorithms Group at Technische Universität Berlin, Germany. The authors thank Dr. Michael Drexl for his comments and support in connection with the use of the boost resource constraint shortest path algorithm.

REFERENCES

[1] T. Achterberg, Scip: Solving constraint integer programs, Math Program Comput 1 (2009), 1–41.
[2] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance, Branch-and-price: Column generation for solving huge integer programs, Oper Res 46 (1998), 316–329.
[3] J.M. Belenguer, M.C. Martinez, and E. Mota, A lower bound for the split delivery vehicle routing problem, Oper Res 18 (2000), 801–810.
[4] G. Bronmø, B. Nygreen, and J. Lysgaard, Column generation approaches to ship scheduling with flexible cargo sizes, Eur J Oper Res 200 (2010), 139–150.
[5] A. Ceselli, G. Righini, and M. Salani, Column generation for the split delivery vehicle routing problem, Technical report, University of Milan - DTI - Note del Polo n. 118, 2009.
[6] G. Desaulniers, Branch-and-price-and-cut for the split delivery vehicle routing problem with time windows, Oper Res 58 (2010), 179–192.
[7] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis, "Time constrained routing and scheduling," in: M. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser (Editors), Network routing, Volume 8, Handbooks in operations research and management science, Chapter 2, Elsevier Science B. V., North-Holland, Amsterdam, 1995, pp. 35–139.
[8] J. Desrosiers and M.E. Lübbecke, "A primer in column generation," in: G. Desaulniers, J. Desrosiers, and M.M. Solomon (Editors), Column generation, Springer-Verlag, Berlin, 2005, pp. 1–32.
[9] M. Dror, G. Laporte, and P. Trudeau, Vehicle routing with split deliveries, Discr Appl Math 50 (1994), 239–254.
[10] F. Hennig, B. Nygreen, M. Christiansen, K. Fagerholt, K.C. Furman, J. Song, G.R. Kocis, and P.H. Warrick, Maritime crude oil transportation—A split pickup and split delivery problem, Eur J Oper Res 218 (2012), 764–774.
[11] S. Irnich and G. Desaulniers, "Shortest path problems with ressource constraints," in: G. Desaulniers, J. Desrosiers, and M.M. Solomon (Editors), Column generation, Springer-Verlag, Berlin, 2005, pp. 33–65.
[12] M. Jin, K. Liu, and R.O. Bowden, A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem, Int J Prod Econ 105 (2007), 228–242.
[13] C. Lee, M.A. Epelman, C.C. White III, and Y.A. Bozer, A shortest path approach to the multiple-vehicle routing problem with split pick-ups, Trans Res Part B 40 (2006), 265–284.
[14] M.D. McKay and H.O. Hartley, Computerized scheduling of seagoing tankers, Nav Res Logist 21 (1974), 255–264.
[15] R. Nersesian, The economy of shipping venezuelan crude to china, Explor Prod: Oil Gas Rev 2 (2005), 78–80.
[16] M. Nowak, Ö. Ergun, and C.C. White, Pickup and delivery with split loads, Trans Sci 42 (2008), 32–43.
[17] UNCTAD. Review of maritime transport 2011, United Nations, New York and Geneva, 2011.
[18] I. Vacca and M. Salani, The vehicle routing problem with discrete split delivery and time windows, In Proceedings of the 9th STRC Swiss Transport Research Conference, September 9, 2009 in Monte Verita Switzerland, 2009.
[19] F. Vanderbeck, A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem, Manage Sci 47 (2001), 864–879.