Geir Brønmo, Bjørn Nygreen & Jens Lysgaard

Column generation approaches to ship scheduling with flexible cargo sizes
Column generation approaches to ship scheduling with flexible cargo sizes

Geir Brønmo & Bjørn Nygreen
Section of Managerial Economics and Operations Research
Norwegian University of Science and Technology
7491 Trondheim
Norway
{geir.bronmo,bjorn.nygreen}@iot.ntnu.no

Jens Lysgaard
Department of Business Studies
Aarhus School of Business
Fuglesangs Allé 4, 8210 Aarhus V
Denmark
lys@asb.dk

Abstract

We present a Dantzig-Wolfe procedure for the ship scheduling problem with flexible cargo sizes. This problem is similar to the well-known pickup and delivery problem with time windows, but the cargo sizes are defined by an interval instead of a fixed value. We show that the introduction of flexible cargo sizes to the column generation framework is not straightforward, and we handle the flexible cargo sizes heuristically when solving the subproblems. This leads to convergence issues in the branch-and-price search tree, and the optimal solution cannot be guaranteed. Hence we have introduced a method that generates an upper bound on the optimal objective. We have compared our method with an a priori column generation approach, and our computational experiments on real world cases show that the Dantzig-Wolfe approach is faster than the a priori generation of columns, and we are able to deal with larger or more loosely constrained instances. By using the techniques introduced in this paper, a more extensive set of real world cases can be solved either to optimality or within a small deviation from optimality.

Keywords: Transportation, integer programming, dynamic programming.
1 INTRODUCTION

Ship scheduling is generally concerned with determining sequences of ports to be visited by ships, taking temporal aspects into account. Many specific types of problems fall into the category of ship scheduling. Due to their inherent complexities, an effective solution approach must invariably be tailored to the particular problem at hand. This involves a decision on an appropriate mathematical model for which we hope to find an optimal or near-optimal solution.

At the level of abstraction corresponding to a mathematical model, ship scheduling may be viewed as being rather similar to vehicle routing and scheduling, representing transportation by sea and road, respectively. The considerable effort in the literature that has been put in attacking vehicle routing and scheduling problems – see e.g. Toth & Vigo (2002) – suggests that a large number of mathematical models and associated solution procedures would be available for being adopted to ship scheduling. However, from a practical perspective, ship scheduling contains certain problem variations which are very rarely encountered in road transportation. Christiansen et al. (2004) recently conducted a thorough survey of ship scheduling research.

The complication that we pay particular attention to in this paper is that the decision about which cargos to carry on a particular ship also involves a decision about the size of each cargo. A decision of cargo sizes requires a balance between profit per unit loaded on the one hand and resource consumption on the other hand, the latter being complicated by time windows in combination with quantity dependent loading/unloading times.

The type of problem that we address here has been introduced by Brønmo et al. (2006), in which the problem was solved as a Set Partitioning Problem (SPP). The approach in their paper is based on a priori generation of all feasible columns, which is reasonable in cases of small or tightly restricted instances. A priori column generation has been the most common optimization method used within ship scheduling research; see for instance Christiansen and Fagerholt (2002), Sherali et al. (1999) and Bausch et al. (1998). On larger or loosely restricted instances where the a priori column generation fails, a dynamic column generation scheme can be used to solve the given SPPs.

In this paper we consider the possibilities for dynamic column generation in ship scheduling with flexible cargo sizes. Our motivation for addressing this issue is that the a priori generation of all feasible columns is intractable when attacking some of the actual problems that we have encountered in practice. We show that the introduction of flexible cargoes to the column generation framework is not straightforward, and therefore we handle the flexible cargo sizes heuristically when solving the subproblem. We have conducted computational experiments on a variety of practical instances in order to compare the performance of our column generation approach to that of the a priori generation of columns in Brønmo et al. (2006).
The rest of this paper is organized as follows: A problem description and mathematical model is presented in Section 2, while the column generation approach is described in Section 3. In Section 4 a computational study is presented and finally, concluding remarks are given in Section 5.

2 PROBLEM DESCRIPTION

The studied short-term ship scheduling problem for the tramp market is closely related to the multi-vehicle pickup and delivery problem with time windows. In the pickup and delivery problem a set of vehicles must satisfy a set of transportation requests having an origin and a destination with time windows and a fixed quantity. In the problem studied here, each transportation request or cargo is given a profit rate, and not all transportation requests must be fulfilled. Some of the cargoes are contracted, and must be carried, while the rest of the cargoes are negotiated on the spot market. Before an agreement is made with the cargo owner, the spot cargoes can be treated as optional.

Each transportation request has a flexible cargo size, i.e. the cargo size is given by an interval. The objective is to maximize the profit. Brønmo et al. (2006) give a mathematical model of this problem. To facilitate the description of the column generation approach in Section 3 we will present a similar model in the following, using the same notation.

2.1 Notation

We have $N$ cargoes indexed by $i$. The loading port of cargo $i$ is associated with the node $i$, while the unloading port is associated with the node $N+i$. Let the fleet of ships be indexed by $v$. Further we define the following sets:

- $\mathcal{N}$: The set of nodes
- $\mathcal{A}$: The set of arcs
- $\mathcal{N}_P$: The set of loading nodes
- $\mathcal{N}_C$: The set of loading nodes that correspond to contract cargoes
- $\mathcal{N}_O$: The set of loading nodes that correspond to optional spot cargoes
- $\mathcal{N}_D$: The set of unloading nodes
- $\mathcal{V}$: The set of available ships
- $\mathcal{N}_v$: The set of feasible nodes for ship $v$
- $\mathcal{A}_v$: The set of feasible arcs for ship $v$

Let $o(v)$ be the artificial origin of ship $v$ and $d(v)$ be its artificial destination. The origin is the latest port visited by ship $v$ before the planning horizon begins, or a given position at sea at the beginning of the planning horizon. The destination is the latest port that will be planned for this ship by solving the model. Now, we define the following constants:

- $Q_{Mi}$: The minimum load quantity for cargo $i$
The maximum load quantity for cargo \( i \)

The sailing time from node \( i \) to \( j \) with ship \( v \)

The time to load or unload one unit of cargo at node \( i \)

The earliest possible time for start of service at node \( i \)

The latest possible time for start of service at node \( i \)

The cost of sailing from node \( i \) to \( j \) with ship \( v \)

The revenue rate of cargo \( i \)

The (possibly negative) profit of using a spot charter vessel for contract cargo \( i \)

(0 for spot cargoes)

The capacity of ship \( v \)

Let \( x_{ijv}, v \in \mathcal{V}, (i, j) \in \mathcal{A}_v \) be the binary flow variable representing the decision whether or not to sail directly from node \( i \) to node \( j \) with ship \( v \). Further, let \( l_{iv}, v \in \mathcal{V}, i \in \mathcal{N}_v \) be the continuous time variable denoting the time for start of service in node \( i \) for ship \( v \). \( q_{iv}, v \in \mathcal{V}, i \in \mathcal{N}_v \) is a continuous variable representing the total load onboard ship \( v \) at the departure from node \( i \), while \( q_{iv}, v \in \mathcal{V}, i \in \mathcal{N}_v \) is a continuous variable denoting the quantity of cargo \( i \) loaded on ship \( v \). Finally, let \( s_i, i \in \mathcal{N} \) be the binary variable representing the decision whether to service a contract cargo by a spot charter for \( i \in \mathcal{N}_C \), and let \( s_i \) be an explicit slack variable for \( i \in \mathcal{N}_O \).

### 2.2 Mathematical model

The short-term tramp ship scheduling problem with flexible cargo sizes can now be formulated as follows:

$$\max \left[ \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}_{pv}} R_{i} q_{iv} - \sum_{v \in \mathcal{V}} \sum_{(i, j) \in \mathcal{A}_v} C_{ijv} x_{ijv} + \sum_{i \in \mathcal{N}_C} \pi_i s_i \right],$$  \hspace{1cm} (2.1)

$$\sum_{j \in \mathcal{N}_{pv} \cup d(v)} x_{ijv} + s_i = 1, \quad \forall i \in \mathcal{N},$$  \hspace{1cm} (2.2)

$$\sum_{i \in \mathcal{N}_v} x_{ijv} - \sum_{i \in \mathcal{N}_v} x_{jiv} = 0, \quad \forall v \in \mathcal{V}, \forall j \in \mathcal{N}_v \setminus \{o(v), d(v)\},$$  \hspace{1cm} (2.3)

$$\sum_{i \in \mathcal{N}_{dp} \cup o(v)} x_{id(v)v} = 1, \quad \forall v \in \mathcal{V}.$$

(2.4)
\[ x_{ijv}(t_{iv} + T_{Qiv}q_{iv} + T_{Sijv} - t_{jv}) \leq 0, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v, \]  \hspace{1cm} (2.6)

\[ T_{MINiv} \leq t_{iv} \leq T_{MAXiv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_v, \]  \hspace{1cm} (2.7)

\[ x_{ijv}(l_{iv} + q_{jv} - l_{jv}) = 0, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v | j \in \mathcal{N}_{p_v}, \]  \hspace{1cm} (2.8)

\[ x_{i,N+j,v}(l_{iv} - q_{jv} - l_{N+j,v}) = 0, \quad \forall v \in \mathcal{V}, (i, N + j) \in \mathcal{A}_v | j \in \mathcal{N}_{p_v}, \]  \hspace{1cm} (2.9)

\[ \sum_{j \in \mathcal{N}_v} Q_{MINi}x_{ijv} \leq q_{iv} \leq \sum_{j \in \mathcal{N}_v} Q_{MAXi}x_{ijv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_{p_v}, \]  \hspace{1cm} (2.10)

\[ q_{iv} \leq l_{iv} \leq \sum_{j \in \mathcal{N}_v} V_{CAPv}x_{ijv}, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_{p_v}, \]  \hspace{1cm} (2.11)

\[ t_{iv} + T_{Qiv}q_{iv} + T_{Sjv}N_{N+i,v} - t_{N+i,v} \leq 0, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}_{p_v}, \]  \hspace{1cm} (2.12)

\[ \sum_{j \in \mathcal{N}_v} x_{ijv} - \sum_{j \in \mathcal{N}_v} x_{j,N+i,v} = 0, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v, \]  \hspace{1cm} (2.13)

\[ x_{ijv} \in \{0,1\}, \quad \forall v \in \mathcal{V}, (i, j) \in \mathcal{A}_v, \]  \hspace{1cm} (2.14)

\[ s_i \in \{0,1\}, \quad \forall i \in \mathcal{N}. \]  \hspace{1cm} (2.15)

Equation (2.1) defines the profit maximizing objective. Constraints (2.2) ensure that all contract cargoes are serviced, and that each optional cargo is not serviced more than once. Constraints (2.3) and (2.5) represent the flow constraints that make sure that the ship leaves the artificial origin and arrives at the artificial destination respectively, and (2.4) are the flow conservation constraints for all other nodes. Constraints (2.6) make sure that the correct arrival times are calculated so that the time window constraints (2.7) can be controlled. The variable \( t_{iv} \) is artificial when ship \( v \) does not visit node \( i \). Constraints (2.6) have to be adjusted slightly for the \( N+i \) nodes, since variable \( q_{iv} \) is defined for pickup nodes (cargoes), only. This is omitted here as in Brønmo et al. (2006). Constraints (2.8) and (2.9) make sure that the load quantities are calculated so that the load quantity intervals (2.10) and the ship capacity constraints (2.11) can be controlled.

The precedence constraints (2.12) force each node \( i \) to be visited before node \( N+i \), while constraints (2.13) ensure that the same ship \( v \) visits both the loading and the unloading node for a
given cargo. Finally, binary requirements are imposed on the flow variables (2.14) and spot charter variables (2.15).

3 COLUMN GENERATION

We introduce a column generation approach to solve the model presented in Section 2. Column generation techniques have been widely used within land-based routing research. Examples of column generation techniques for solving pickup and delivery type problems are given by Dumas et al. (1991), Ioachim et al. (1995), Xu et al. (2003) and Sigurd et al. (2004). Within ship scheduling this approach has not been seen so often. Appelgren (1969) used column generation to a somewhat simpler ship scheduling model, while Christiansen and Nygreen (1998a and 1998b) and Persson and Göthe-Lundgren (2005) used column generation to solve marine inventory routing problems.

In our column generation approach the problem is decomposed into a master problem with a promising collection of ship schedules or columns and a subproblem for each ship where the promising ship schedules are found. The master problem is a set partitioning problem that will be presented in Section 3.1. In Section 3.2 an overview of the column generation scheme is given. The subproblem is a shortest path problem with pickup and delivery, time windows and flexible cargoes. The subproblem is described in Section 3.3, while Section 3.4 describes a solution approach for the subproblem with fixed quantities. Sections 3.5 and 3.6 describe several ways of handling the flexible cargoes. A branch-and-price algorithm that finds good integer solutions to this problem is presented in Section 3.7 while we present a method for calculating an upper bound to the optimal objective in Section 3.8.

3.1 The master problem

Column generation techniques are based on the Dantzig-Wolfe decomposition algorithm described by Dantzig and Wolfe (1960). In their algorithm the problem is reformulated into a master problem and several LP subproblems. The master problem contains the objective and common restrictions that are expressed by convex combinations of subproblem solutions. The optimal solution is a convex combination of subproblem extreme points. In the Dantzig-Wolfe approach the extreme points are generated dynamically in an iterative process. In many problems related to vehicle routing an integer combination of subproblem solutions is required and the master problem becomes a set partitioning problem. Here, we use a set partitioning reformulation of the original problem where the columns are dynamically generated by iteratively solving the subproblems. For the sake of simplicity, and since the mathematical models of the subproblems are equal, in the following we will refer to the set of subproblems as the subproblem. Brønmo et al. (2006) gave a set partitioning reformulation of the tramp ship scheduling problem with flexible cargoes. We will use a similar formulation as our master problem.
Let $\mathcal{R}_v$ be the set of all feasible cargo sets for ship $v$, indexed by $r$. For each cargo set there is an optimal schedule that is defined by a sequence of port calls with arrival times and load quantities. Define the profit of the optimal schedule of each cargo set as $P_{vr}$. Further, let $A_{ivr}$ be a constant equal to the number of times cargo $i$ is serviced by ship $v$ in the sequence of cargo set $r$. The binary variables $y_{vr}$ denote whether cargo combination $r$ is chosen for ship $v$ or not. The set partitioning formulation of the tramp ship scheduling problem with spot charters can now be given as follows:

\[ \begin{align*}
\text{max} & \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} P_{vr} y_{vr} + \sum_{i \in \mathcal{N}_C} \pi_i s_i , \\
\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} A_{ivr} y_{vr} + s_i = 1, & \forall i \in \mathcal{N}, \\
\sum_{r \in \mathcal{R}_v} y_{vr} = 1, & \forall v \in \mathcal{V},
\end{align*} \]  

(3.1)

(3.2)

(3.3)

(3.4)

(3.5)

Equation (3.1) defines the maximizing profit objective. Constraints (3.2) ensure that all contract cargoes are serviced, and that each optional cargo is not serviced more than once. Constraints (3.3) ensure that each ship in the fleet is assigned exactly one column (represented by a schedule or slack), while constraints (3.4) and (3.5) impose the binary requirements on the variables.

It should be noted that in the case of cycles we have at least one $A_{ivr} > 1$. Cycles can occur if the subproblem is relaxed so that the sequences need not be elementary, i.e., it is allowed to load a cargo more than once. A column that contains a cycle will never be a part of a feasible integer solution to the master problem due to the variables $y_{vr}$ and the right-hand side of equations (3.3).

To facilitate the description of the column generation scheme below, we need the following definitions:

ILP-master: (3.1) – (3.5).

LP-master: (3.1) – (3.3), $y_{vr} \geq 0$, $\forall v \in \mathcal{V}, r \in \mathcal{R}_v$, $s_i \geq 0$, $\forall i \in \mathcal{N}$.

Restricted master: LP-master, where $\mathcal{R}_v$ is replaced by $\mathcal{R}_v' \subseteq \mathcal{R}_v$. 


3.2 Column generation overview

If all feasible cargo set and ship combinations were known a priori, then solving the ILP master would give the optimal solution to the problem. In our column generation scheme only a small subset of the cargo set – ship combinations or columns are generated in advance. We start out by solving the restricted master defined above with the given columns. Now, the values of the dual variables corresponding to equations (3.2)-(3.3) are transferred to the subproblem. The optimal solution of the subproblem must have a non-negative reduced cost, since the basic columns of the restricted master problem represent feasible subproblem solutions with reduced cost 0.

An optimal subproblem solution can be categorized as follows: If the reduced cost is positive, the solution represents a new column that might lead to a better solution. On the other hand, if the reduced cost is 0, the column is either known in advance or is not wanted into the basis in this iteration since it will not improve the restricted master solution. Solving the subproblem might give a set of new columns that have a positive reduced cost. The new columns are added to the restricted master, and the process continues iteratively until no more positive reduced cost columns can be found. Now, we have the optimal solution to the LP-master. If the solution is integer, it is also the optimal solution of the ILP-master. Otherwise, branch-and-bound search with column generation in every node (often referred to as branch-and-price) is used to find the optimal ILP-master solution. For an overview of branch-and-price methods and applications see Barnhart et al. (1998).

3.3 The subproblem

The ILP-master ensures that constraints (2.2) and (2.15) of the original model in Section 2 are satisfied. When these constraints are removed from the model, the problem decomposes into one subproblem for each ship. The dual variables of constraints (3.2)-(3.3) are introduced into the objective function of the subproblem such that each solution with a positive objective value represents a dual infeasible solution – and hence a promising column – to the restricted master problem.

Let $u_i$ be the dual variable for the constraint of type (3.2) representing cargo $i$ in the restricted master problem, and let $w_v$ be the dual variable associated with the constraint of type (3.3) representing ship $v$. Now, associate $\sigma_i = u_i$ to the loading node and $\sigma_{N+i} = 0$ to the unloading node for all cargoes $i \in N_{Pv}$. Finally associate $\sigma_o = w_v$ to the origin $o(v)$ of ship $v$. The subproblem for ship $v$ can now be formulated as follows:

$$\max \left[ \sum_{i \in N_{Pv}} P_i q_{iv} - \sum_{(i,j) \in A_v} (C_{ijv} + \sigma_j) x_{ijv} \right],$$

s.t. (2.3)-(2.14)
Equation (3.6) represents the objective function of the subproblem. If the objective of the optimal subproblem solution is positive, this solution represents a column that might improve the restricted master solution and is wanted in the basis. If the optimal objective of the subproblem is nonpositive for all \( v \in \mathcal{V} \), no improving columns can be found, and we have an optimal solution to the LP-master.

The subproblem can be described as the problem of finding the optimal schedule for the given ship under the given restricted master dual variable values. The schedule is a discrete sequence from origin through some of the nodes to the artificial destination containing arrival times and load quantities. This problem is a shortest path problem with pickup and delivery, time windows and flexible cargoes. This is clearly an NP-hard problem, since it is a generalization of the NP-hard shortest path problem with time windows (SPPTW), see for instance Desrosiers et al. (1995).

In the problem formulation, time and quantity are given as continuous variables. If time and quantity were discretized, the subproblem would be easier to solve, but the potential number of nodes in the network would be very large. Several authors have shown that it is possible to handle continuous time by using dynamic programming to solve the subproblem, see for instance Dumas et al. (1991), Christiansen and Nygreen (1998b) and Sigurd et al. (2004). However, handling continuous load quantities of non-homogeneous commodities within a dynamic programming approach is not straightforward. In a dynamic programming approach partial solutions are typically represented by labels. Since the solution space and hence the label space is typically large, the efficiency of such methods relies on the elimination of dominated labels. In the fixed cargo sizes case, each label consists of the objective value, arrival time and the cargoes that are currently loaded onboard the ship. Now, the arrival time and objective values of a label can be compared if they have the same set of cargoes on board. In the flexible cargo cases we would need to introduce the quantity loaded of each cargo on board into the label. However, as we will show later, such labels cannot be made if we want to keep one label for each sequence, since it is not possible to determine the optimal load quantity of the cargoes in a sequence before the whole sequence is known. It is therefore impossible to eliminate labels since the labels cannot be compared, and the method becomes computationally intractable.

In order to make a method that can solve practical instances, we use discrete load quantities when solving the subproblem. The load quantity interval of each cargo is divided into a small number of alternative discrete quantities. As we will describe in Section 3.6, we find the optimal load quantities before a column is added to the restricted master, so that the objective value is as close as possible to the correct one.

Since we use discrete load quantities, the subproblem can be solved with slightly modified dynamic programming methods known from the literature. We have used the method of Dumas et al. (1991) as a starting point.
3.4 Solving the subproblem with fixed cargo sizes via dynamic programming

In this section we describe how the subproblem can be solved when the cargo sizes are fixed, i.e. when we have only one single load quantity per cargo. The subproblem can now be easily reformulated into a shortest path problem with pickup and delivery and time windows (SPPDTW). Dumas et al. (1991) presented a column generation approach for solving the multi-vehicle pickup and delivery problem with time windows (m-PDPTW). We will use a slightly enhanced version of their algorithm to solve the subproblem with fixed cargo sizes.

First, we describe how the subproblem with fixed cargo sizes can be reformulated to a SPPDTW. Since the cargo sizes are fixed, the profit of servicing a cargo can be incorporated into the arc cost of all arcs going into the loading node. Call these updated cost coefficients $\bar{c}_{ijv}$. In addition, define $\bar{c}_{ijv} = c_{ijv} + \sigma_j$. The objective of the subproblem with fixed cargo sizes can now be formulated as follows:

$$\max_{v} \sum_{(i,j) \in A_v} -\bar{c}_{ijv}x_{ijv},$$

(3.7)

Constraints (2.6) of the original model can in this case be modified, since service times and hence the total time between nodes can also be calculated from the load quantities. Finally, constraints (2.10) can be removed from the model.

In line with Dumas et al. (1991), we solve the SPPDTW via a forward dynamic programming algorithm. In order to outline the algorithm, we need to define labels that each represents a unique sequence of nodes that may be extended to be the optimal solution.

Let $l$ be a node in the graph of the subproblem and let $k$ be a feasible sequence of nodes beginning at $o(v)$ and ending at node $l$.

- $z_{lk}$ is the objective value of sequence $k$
- $a_{lk}$ is the arrival time at node $l$ of sequence $k$
- $S_{lk}$ is the set of cargoes that are onboard the ship at node $l$ of path $k$

Each label needs to store some information that can help restore the final solution when the algorithm has terminated. As discussed by Dumas et al. (1991), there are two alternatives: Either a reference to the preceding label must be stored, or we need to store the whole visit sequence. In the first alternative all non-dominated labels must be kept in memory, while in the second only two iterations of labels need to be kept. However, in the second alternative the size of a label grows with the length of the sequence. What is the most efficient of the two alternatives probably depends on the average length of the sequences. In line with Dumas et al. (1991) we have chosen the second alternative.
The algorithm can be outlined as follows:

Start with the node $o(v)$ with the objective $z_{lk} = 0$, $a_{lk} = T_{MN_o(v)}$.

Until no more labels can be made, do:

For all labels made in the previous iteration, do:

Make a new label for each feasible extension of the sequence

Calculate arrival time and objective value

If the ship is empty, we have a feasible solution – update the set of best solutions so far

Delete all labels that cannot be a part of the optimal solution (dominated labels)

The algorithm starts with the sequence containing the start node $o(v)$ of ship $v$. Then, in each iteration the algorithm considers all labels that were made in the previous iteration and creates a new label for each feasible extension. The label extensions are trivial. Let $T_{T_{ij}}$ be the time from start of service of node $i$ to the arrival at node $j$. The extension from node $l$ and sequence $k$ to node $n$ and sequence $m$ can now be described as follows: If $n \in S_{lk}$, we have an unloading node and $S_{nm} = S_{lk} \cup \{n\}$. Otherwise, we have a loading node and $S_{nm} = S_{lk} \setminus \{n\}$. The updating of the arrival time and objective values are given by the following equations:

$$z_{nm} = z_{lk} - \bar{C}_{lnv},$$

$$a_{nm} = \max(a_{lk} + T_{T_{ln}}, T_{MNn})$$

The algorithm terminates when no more labels can be created. The optimal solution is represented by the label with empty $S_{lk}$ that has the largest value of $z_{lk}$. We have implemented this algorithm so that we store the $p$ best solutions during the process, where $p$ is a parameter to the algorithm. We also store candidates with negative objective values. Such solutions are not needed at this point, but later, quantity optimization at the end might make negative reduced cost solutions positive. When the algorithm is terminated, all the solutions among the $p$ best with positive objective value represent promising columns and are added to the restricted master.

This algorithm does not guarantee that the optimal solution is elementary, i.e. that it contains no cycle. However, since a cargo must be lifted twice in order to generate a cycle, we can state that cycles are scarce unless the time windows are wide. In this paper, cycles will be eliminated in the branch-and-bound search.

Dumas et al. (1991) present four propositions for label elimination that are crucial to the performance of the algorithm. The first two propositions concern the feasibility of the sequence given by the label:
Proposition 1. A label \((z_{lk}, a_{lk}, S_{lk})\) such that \(i \in S_{lk}\) is eliminated if the sequence extension \(l \rightarrow N+i\) is infeasible.

Proposition 2. A label \((z_{lk}, a_{lk}, S_{lk})\) such that \(i, j \in S_{lk}\) is eliminated if both sequence extensions \(l \rightarrow N+i \rightarrow N+j\) and \(l \rightarrow N+j \rightarrow N+i\) are infeasible.

The first two propositions could be extended to include all the members of the set \(S_{lk}\), but this is computationally intractable. Hence, only subsets of \(S_{lk}\) of cardinality 1 and 2 are checked for infeasibility.

The final two propositions concern the elimination of dominated labels:

Proposition 3. If two labels representing the sequences \(k\) and \(m\) both ending at node \(l\) are such that \(S_{lk} = S_{lm}\), \(a_{lk} \leq a_{lm}\) and \(z_{lk} \geq z_{lm}\), the label \((z_{lm}, a_{lm}, S_{lm})\) can be eliminated.

The proof of this proposition is based on the fact that every feasible extension of sequence \(m\) is feasible for the sequence \(k\).

Proposition 4. If two labels representing the sequences \(k\) and \(m\) are such that \(S_{lk} \subseteq S_{lm}\), \(a_{lk} \leq a_{lm}\) and \(z_{lk} \geq z_{lm}\), and if the cost values \(C_{ijv}\) satisfy the triangle inequality, the label \((z_{lm}, a_{lm}, S_{lm})\) can be eliminated.

The proof is given by Dumas et al. (1991), and is based on the fact that when the node \(j\) is a delivery node, we know that \(\bar{C}_{ijv} = C_{ijv} = C_{ijv}\).

Propositions 3 and 4 are very important for the computational efficiency of the algorithm. If these propositions are not used, only infeasible labels will be eliminated. Hence, the algorithm would perform a full enumeration of sequences with the additional possibility of generating cycles.

In our code we have extended the use of proposition 4. Define the cardinality difference between two labels that satisfy this proposition, \(d = |S_{lm}| - |S_{lk}|\). Dumas et al. (1991) use \(d = 1\), while we set the maximum value of \(d\) to be a parameter in our algorithm.

3.5 The subproblem with discretized load quantities

We will in this section describe how the subproblem can be solved with discrete load quantities. We introduce a set of discretization nodes to represent the loading of cargo \(i\). A discretization loading node for cargo \(i\) is a copy of the loading node of the cargo. Each node in the set corresponds to a fixed load quantity in the interval \([Q_{MINi}, Q_{MAXi}]\). Let the parameter \(b\) specify the number of discretization nodes for each cargo. For instance if \(b = 3\), the three loading quantities for
cargo $i$ are set at $Q_{MNi}$, $(Q_{MNi} + Q_{MXi})/2$ and $Q_{MXi}$. The $T_{ijv}$ and $C_{ijv}$ values are recalculated for the given quantity. Some of the arcs might be disregarded due to infeasibility.

The algorithm presented in Section 3.4 must be slightly modified in order to solve the subproblem with a given discretization. We now define a label for sequence $k$ ending at node $l$ as follows:

$$z_{lk} \text{ and } a_{lk} \text{ as in Section 3.4}$$

$S_{lk}$ is the set of loading nodes that are onboard the ship at node $l$ of sequence $k$

In the discretization case it is infeasible to extend a label to any of the loading nodes corresponding to any of the cargoes that are on board the ship. With this modification the subproblem is solved to optimality for the given discretization.

The potential number of labels compared to the fixed quantity case is large, and grows exponentially with the number of discretization nodes, so this approach may become computationally intractable even with a small number of discretization nodes. To reduce the complexity we have implemented an additional algorithm that we call approximated discretization. Now, the label definitions are the same as in Section 3.4. Hence, a label does not specify the quantities of the cargoes on board. To ensure optimality with the given discretization, two labels with equal $S_{lk}$ cannot be directly compared. The label with the lowest objective value might still have higher profit potential than the other if the quantities loaded are smaller. The domination criteria of propositions 3 and 4 are no longer valid. However, we still use the criteria and therefore the approximated discretization method becomes heuristic even for the given discretization.

### 3.6 The subproblem with flexible cargo sizes – finding load quantities by LP

To the authors’ knowledge no algorithm is available that solves the SPPDTW with flexible cargo sizes to optimality within reasonable computation time. In this section we show that the method from Section 3.4 cannot easily be extended to the flexible case, and we propose a heuristic way to handle the flexible cargo sizes that can be used for any discretization of the cargo sizes.

The method in Section 3.4 is based on the comparison of labels that represent partial solutions arriving at the same node with a correct objective value and a correct arrival time and the set of cargoes on board the ship. With the introduction of continuous non-homogeneous cargo sizes we also need to store the quantity loaded of each cargo in the label. It is impossible to generate correct labels of this type, since the optimal load quantities are not unique for cargoes in an unfinished sequence, i.e. when the ship is not empty at the last node. Consider the small example given in Figure 1. Two cargoes are given, cargo $i$ with $Q_{MNi} = 4$, $Q_{MXi} = 6$, $R_i = 3$ that is picked up in node $i$ and delivered in node $N+i$ and cargo $j$ with $Q_{MNj} = 4$, $Q_{MXj} = 6$, $R_j = 4$ that is picked up in node $j$ and delivered in node $N+j$. $V_{CAPv} = 10$. For the sake of simplicity we assume that the time constraints are not binding. Consider the sequence $i$ that might be extended to the sequences $i-j$ and $i-(N+i)$. For the extension to sequence $i-j$ the optimal load quantity of cargo $i$, $q_i^* = 4$ since cargo $j$ is more
profitable, while for the extension to sequence $i-(N+i)$, $q_i^* = 6$. This means that the optimal load quantity of the label for sequence $i$ is not unique and cannot be found as long as we do not allow several labels per sequence.

$$V_{CAP} = 10$$

$q_i, q_j \in [4, 6]$}

$R_i = 3$

$R_j = 4$

$q_i^* = 6$

Figure 1 Small quantity optimization example

The correct load quantities can only be established for labels where $S_{lk}$ is empty, i.e. in nodes where the ship is empty. Hence, the elimination of dominated labels becomes difficult in the cases where $S_{lk}$ is not empty. Proposition 3 (and hence proposition 4) is based on the fact that in the fixed cargo sizes case, when time, objective value and the set of cargoes are identical, the possible label extensions are identical. Now we assume that we calculate the load quantities for two labels with unfinished sequences that otherwise fulfil proposition 3. Possible label extensions can alter the quantity decisions, so the objective values and arrival times would not be correct and could not be compared. It is also difficult to say anything about the feasibility of future label extensions when the load quantities are not known. Hence, propositions 3 and 4 are not valid in the continuous load quantity case.

In order to establish a simple optimization algorithm, we might use a different approach: First, we disregard the domination criteria of propositions 3 and 4. Second we fix all cargo sizes at minimum. For all labels with empty $S_{lk}$, we calculate the optimal load quantities and compare the resulting objective against the $p$ best solutions. However, further extensions of the label must be based on the objective calculated with minimum quantities. Otherwise the algorithm is unchanged. Then, the optimal solution is found, since all label comparisons are based on correct objective values. However, this approach is likely to be computationally intractable, since the elimination of dominated labels is crucial for the performance of the algorithm. As mentioned in Section 3.3, this method will be less efficient than a full enumeration of sequences.
Since the proposed optimization algorithm is likely to be computationally intractable, we have chosen a heuristic way of handling the flexible cargo sizes.

Our heuristic approach is based on using the domination criteria of propositions 3 and 4, even if labels leading to the optimal solution might be disregarded. Our approach for solving the subproblem can be described as follows: First, the quantity of each cargo is either fixed at its minimum value or discretized as described in the previous section. Second, we use the dynamic programming algorithm presented in Section 3.5 to find the $p$ best solutions to the subproblem during the column generation process. Then, when the dynamic programming algorithm has terminated, we find the optimal load quantities for each of the $p$ best solutions. Brønmo et al. (2006) showed that the problem of finding the optimal load quantities for a given sequence can be formulated as an LP, and introduced an efficient preprocessing and solution algorithm. We use their solution algorithm to find the optimal load quantities for a given sequence. As opposed to the sequences in their paper, our subproblem solutions might have cycles. Since cycles are allowed in their load quantity LP model, the method needs no modification.

After quantity optimization those subproblem solutions among the $p$ best with positive objective values are introduced to the restricted master. The objective coefficient value of the new columns in the restricted master is calculated from the optimal load quantities.

In our approach the objective coefficient values of the restricted master columns are heuristic, since there might be an unidentified sequence for the given cargo set – ship combination with higher objective value. In the dynamic programming algorithm, quantities are discrete or fixed at minimum. Hence, a label neither contains the correct objective value nor the correct time or total quantity loaded. This way there is a possibility that the label that eventually leads to the optimal subproblem solution might be disregarded due to dominance. And even if it was not dominated, the unidentified optimal solution is not guaranteed to be among the $p$ best solutions. Hence, the best solution from solving the subproblem might have negative reduced cost, something that will never occur in a standard Dantzig-Wolfe approach. The column generation scheme will potentially converge before the optimal solution of the LP-master is found.

### 3.7 Finding an integer solution by branch-and-price

When the column generation process converges, no more potentially improving columns can be found. In the cases where the subproblem is solved to optimality we have an optimal solution to the LP-relaxed master problem. If the solution is fractional, we use a branch-and-price procedure to find the optimal solution to the problem.

In a branch-and-price method, the problem is divided into two different problems by branching in each node. We have chosen to branch on a cargo-ship combination. Here, we choose a cargo and ship combination to branch on. In the 1-branch all columns for the given ship are to contain the given cargo, and no columns for the other ships may contain this cargo. In the 0-branch no column
may contain the given cargo-ship combination. By choosing this way of branching, no change is
needed in the subproblem solution procedure. We only change the temporal data in a given node,
so that in the 1-branch the given cargo will only be feasible (and extra profitable to enhance
efficiency) for the given ship. In the 0-branch the cargo is made infeasible for the given ship.

Since we do not eliminate cycles when solving the subproblem, branching on cargo-ship
combinations is not sufficient to ensure an integer solution. A column that contains cycles has at
least one $A_{ijr}$ greater than one. If such a column has a positive $y_{ijr}$ value in the optimal LP-master
solution, then $y_{ijr}$ is fractional due to constraints (3.3). We use time window branching if at least
one of the columns in the solution of the current node has a cycle. Different variants of time
window branching have been used by for instance Christiansen and Nygreen (1998a) solving a
marine inventory routing problem and Gelinas et al. (1995) solving a vehicle routing problem. A
cycle can be identified as a cargo being serviced at least twice by the same ship. Now, the time
window of the given cargo is divided into two time windows so that the cycle becomes infeasible in
both. In one branch the earlier of the two time windows is used, while in the other branch the later
time window is used.

For the sake of simplicity of implementation, and to find a feasible solution quickly, we have
chosen a depth-first search strategy, where we follow the branch with the early time window first.

In a given node in the search tree with a fractional solution, we use the following procedure to
select the type of branching to use, and what entity to branch on:

IF there are cycles in the solution
  Choose time window branching.
  Select one of the cycles. The cycle is characterized by at least two arrivals at the loading
  node of cargo $i$. Calculate the average of the earliest and latest arrivals at the loading node
  in the given sequence and choose it as the branching time.

ELSE
  Choose cargo-ship branching.
  Calculate the solution weight of each cargo-ship combination. The solution weight of a
cargo-ship combination is defined as the sum of the values of the basis variables that
contain this combination. Choose the cargo-ship combination for which the solution weight
is closest to 0.5.

In the cases where the subproblem is not solved to optimality, the final branch-and-price solution is
not guaranteed to be optimal. Normally, the node solution value will be used as an upper bound to
the given node problem, but this is not true in the heuristic case. It is furthermore possible that an
improving column that was not generated in the root node might be found in a later node of the
search giving a rise in the node value. Hence, the node values do not necessarily decrease
monotonously with the tree depth. This means that if the node value is lower than the best solution
found so far, it might not be correct to backtrack. Anyway we backtrack in such a situation and the branch-and-price procedure might converge ‘too early’.

### 3.8 Upper bound

We find an upper bound to the problem presented in Section 2 by discretization. We will use this bound to benchmark the quality of the heuristic methods presented earlier.

We use the discretization approach from Section 3.5 as a starting point, where the loading of cargo \( i \) is represented by a set of nodes where each node \( k \) is given a fixed quantity \( Q_k \in [Q_{MIN}, Q_{MAX}] \) that represents the loading of cargo \( i \). In a set of discretization nodes, let the next node \( l \) of node \( k \) be defined as the node where \( Q_l \) is closest to \( Q_k \) and \( Q_l > Q_k \). Now, we define a new model that we call upper bound discretization. In the original model, each loading node is associated with a load quantity and a loading time, unloading time and income. In the upper bound discretization model, the quantity, loading time and unloading time of each loading node are kept the same while the income is increased. Each loading node \( k \) is given the income of its next node \( l \). This way the income is higher, while the resource consumption and hence the costs are the same as in the original model. The ‘maximum’ node, i.e. the discretization node \( k \) of cargo \( i \) with \( Q_k = Q_{MAX} \) is omitted here, since it has no next node.

Consider cargo \( i \) of the example considered earlier (Figure 1) where \( q_i \in [4, 6] \) and \( R_i = 3 \). If the quantity of cargo \( i \) is discretized so that \( q_i \in \{4, 5, 6\} \), the set of incomes for the nodes would be \( \{12, 15, 18\} \). The corresponding upper bound discretization for cargo \( i \) would be \( q_i \in \{4, 5\} \) with the set of incomes \( \{15, 18\} \).

When there are only two nodes in the upper bound discretization model, this approach is equivalent to solving the model with quantities fixed at minimum and the income at maximum. However, we believe that this will not be accurate enough to give a tight bound. Therefore we have allowed for a small user specified number of upper bound discretization nodes in our code.

We solve the upper bound discretization model to optimality using the branch-and-price approach previously presented with the subproblem solution method from Section 3.5 (without quantity optimization). The optimal solution of the upper bound discretization model is an upper bound to the original problem. Any feasible solution to the original problem can be changed to an upper bound discretization solution that consumes less or the same amount of time and capacity resources and has equal or higher objective value. This can be done by rounding down the load quantity of each cargo to the quantity of the nearest discretization node and by rounding up the income of each cargo to the income from the nearest discretization node. This way the optimal solution of the upper bound discretization approach will never have a lower objective value than the optimal solution of the original problem.
4 COMPUTATIONAL RESULTS

We have tested our solution approach on ten test cases based on real data from the tramp shipping industry. The data are collected from two different shipping companies. The motivation was to evaluate the approach in terms of response time and solution quality.

The test cases are described in Section 4.1. In Section 4.2 parameter values are found, while the computational results are presented in Section 4.3.

4.1 Case descriptions

Cases 1 to 5 are collected from a shipping company operating in northern Europe, transporting dry bulk commodities such as rock, iron ore and cement. Cases 1 to 4 are based on scheduling problems for a fleet of four ships. The time windows are large for both loading and unloading of the cargoes. In case 5 the fleet is increased to six ships and the time windows are made somewhat narrower.

Cases 6 to 10 are represented by a chemical commodity shipping company operating between Europe and the Caribbean. In these cases the number of ships varies from 3 to 7. The ships come from the same fleet. In cases 6, 7 and 9 some ships are engaged in fulfilling other obligations, while in cases 8 and 10 the whole fleet is available. For cases 6, 8 and 9 the time windows are typically narrow for the loading and wide for the unloading of the cargoes, while the unloading time windows are made somewhat narrower for cases 7 and 10.

Cases 1 to 3 and 6 to 8 correspond to cases 2 to 4 and 6 to 8 of the paper by Brønmo et al. (2006). The case descriptions are presented in Table 1.

<table>
<thead>
<tr>
<th>Cases</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning horizon in days</td>
<td>15</td>
<td>15</td>
<td>17</td>
<td>20</td>
<td>17</td>
<td>120</td>
<td>75</td>
<td>75</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td># of cargoes</td>
<td>14</td>
<td>17</td>
<td>17</td>
<td>24</td>
<td>31</td>
<td>15</td>
<td>12</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td># of ships</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

4.2 Finding parameter values

In order to choose sensible values for parameters \(d\) and \(p\) we have performed tuning experiments on cases 2, 6 and 9. In these experiments the subproblem was solved with all quantities at minimum \((b=1)\), and quantity optimization was performed on all the \(p\) best subproblem solutions.
The parameter $d$ concerns the function of proposition 4 of the dynamic programming method described in Section 3.4. It denotes the maximum cardinality difference $|S_{lm}|-|S_{lk}|$ between two labels that satisfy proposition 4. Table 2 shows the effect of changing parameter $d$ on the given cases. The setting $d=0$ is the same as not using proposition 4, while setting $d=1$ is the one used by Dumas et al. (1991). Table 2 shows that the efficiency of the method increases at $d=2$ for cases 6 and 9, but a further increase in $d$ does not affect the CPU times. On the basis of this we have chosen to use $d=3$ in the following.

The parameter $p$ denotes the number of best solutions to keep during the dynamic programming algorithm. It is reasonable to assume that this parameter affects the CPU time of the column generation approach, since it controls the number of promising columns that are transferred to the restricted master each time the subproblem is solved. When quantity optimization is introduced, the column generation convergence and hence the final objective value might also be affected by the value of $p$. The higher the value of $p$, the higher is the probability that the best possible column is among those transferred to the restricted master. Our experiments with case 9 showed that in only 16% of the subproblem runs, the best solution with fixed quantities was still the best after quantity optimization. For cases 2 and 6 this measure was 27% and 38%, respectively.

Table 2 The effect of parameter $d$, CPU times in seconds

<table>
<thead>
<tr>
<th>Cases</th>
<th>2</th>
<th>6</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of $d$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>13</td>
<td>443</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>13</td>
<td>453</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>141</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
<td>141</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>141</td>
</tr>
</tbody>
</table>

Figure 2 shows how the final objective value and the CPU time needed for solving the first node vary with the value of $p$ for case 9. The objective value is given as the gap from the best solution found so far. We use the CPU time for the first node, since it measures the efficiency of the column generation process as such. The number of nodes in the branch-and-bound tree seems to vary more arbitrarily with the value of $p$.

Figure 2 shows that the objective gap decreases with the value of $p$ and that the first node CPU decreases until they both stabilize around $p = 50$ for case 9. A similar picture can be seen from the other cases, but the CPU times increase slightly for the higher $p$ values. On the basis of these results we have chosen to set $p = 60$ in the following.
We also performed tests on the same cases to decide the value of the parameter $b$ that specifies the number of discretization nodes. The parameter settings used are called $LP_b$, so that for instance the setting with one discretization nodes is called $LP_1$. In all settings quantity optimization is performed on all the $p$ best solutions from the subproblem. The results are shown in Table 3.

Table 3 The effect of the number of discretization nodes

<table>
<thead>
<tr>
<th></th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$LP_1$</td>
<td></td>
</tr>
<tr>
<td>Objective gap</td>
<td>0</td>
</tr>
<tr>
<td>CPU</td>
<td>4</td>
</tr>
<tr>
<td>$LP_2$</td>
<td></td>
</tr>
<tr>
<td>Objective gap</td>
<td>0</td>
</tr>
<tr>
<td>CPU</td>
<td>9</td>
</tr>
<tr>
<td>$LP_3$</td>
<td></td>
</tr>
<tr>
<td>Optimality gap</td>
<td>0</td>
</tr>
<tr>
<td>CPU</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 3 shows that there was no effect on the objective value from increasing the number of discretization nodes for the given cases. For cases 2 and 6 the optimal solution was found already with $b = 1$. However the CPU times increase substantially. Case 9 could not be solved with three discretization nodes. On the basis of this we will not use the $LP_3$ setting in the following. We will use both the other settings, since $LP_2$ might give better solutions in the cases where the $LP_1$ solution is not optimal.
4.3 Computational results

The tests were performed on a PC with a Pentium IV, 3.2 GHz processor and 3.0 GB RAM under Windows XP. The column generation method was developed in C++. The LP models for finding optimal load quantities and the master problem model were solved by XpressMP (www.dashoptimization.com).

In Table 4 we compare the results of the column generation approach with the results of the a priori generation approach of Brønmø et al. (2006) in terms of solution quality and computational efficiency. For cases 1-3 and 6-8, “Gap from best solution” means optimality gap, since for these cases we can compare with the optimal solution from the a priori column generation algorithm. In the following we will refer to these cases as the benchmarked cases. In the upper bound version we also have two discretization nodes (minimum with medium income and medium with maximum income). The upper bound gap is given as the gap between the best of the $LP_1$ and $LP_2$ solutions and the upper bound. The CPU times are given in seconds.

<table>
<thead>
<tr>
<th>Cases</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LP_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gap from best solution</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>0.2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>B&amp;B nodes</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>256</td>
<td>1</td>
<td>8</td>
<td>26</td>
<td>30</td>
<td>28</td>
</tr>
<tr>
<td>CPU</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>250</td>
<td>377</td>
<td>3</td>
<td>1</td>
<td>25</td>
<td>120</td>
<td>53</td>
</tr>
<tr>
<td>$LP_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gap from best solution</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B&amp;B nodes</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>218</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>CPU</td>
<td>18</td>
<td>9</td>
<td>6</td>
<td>6630</td>
<td>2518</td>
<td>55</td>
<td>12</td>
<td>157</td>
<td>1919</td>
<td>946</td>
</tr>
<tr>
<td>Upper bound</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper bound gap</td>
<td>1.5</td>
<td>0.04</td>
<td>2.7</td>
<td>3.0</td>
<td>3.0</td>
<td>0.5</td>
<td>1.5</td>
<td>0.4</td>
<td>0.7</td>
<td>2.4</td>
</tr>
<tr>
<td>B&amp;B nodes</td>
<td>6</td>
<td>50</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>12</td>
<td>10</td>
<td>100</td>
<td>14</td>
</tr>
<tr>
<td>CPU</td>
<td>39</td>
<td>50</td>
<td>10</td>
<td>16262</td>
<td>284</td>
<td>40</td>
<td>24</td>
<td>763</td>
<td>6073</td>
<td>1129</td>
</tr>
<tr>
<td>Upper bound gap 1. node</td>
<td>6.1</td>
<td>0.6</td>
<td>2.7</td>
<td>3.0</td>
<td>3.1</td>
<td>0.5</td>
<td>4.0</td>
<td>0.6</td>
<td>1.81</td>
<td>2.4</td>
</tr>
<tr>
<td>CPU</td>
<td>13</td>
<td>12</td>
<td>6</td>
<td>14189</td>
<td>91</td>
<td>40</td>
<td>8</td>
<td>173</td>
<td>1453</td>
<td>544</td>
</tr>
<tr>
<td>A priori column generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B&amp;B nodes</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPU</td>
<td>280</td>
<td>340</td>
<td>53</td>
<td>-</td>
<td>-</td>
<td>2492</td>
<td>64</td>
<td>5542</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4 shows that our approach is much faster than the a priori column generation. Cases 4, 5, 9 and 10 could not be solved by a priori column generation, but high quality solutions were found by our approach. In case 9 the optimality gap is not greater than 0.7%, while in case 4 the upper bound
is 3.0% above the best solution found. The optimal solution was found in all the six benchmarked cases. The $LP_1$ setting is the faster of the two settings tested and provided the optimal solution to three of the cases. The $LP_2$ setting is clearly slower than $LP_1$, but the optimal solution was found in all the six benchmarked cases and the best solution in the other four cases. The $LP_2$ setting was still much faster than the a priori column generation approach. The upper bound version gave bounds from 0.04% to 1.5% above the optimal solution for the six benchmarked cases. This indicates that the real optimality gap of the other four cases might be lower than the upper bound gap. It can be seen from Table 4 that the upper bound version is on average slower than the $LP_2$ setting even though the two approaches appear to be structurally identical. However, it should be noted that the two discretization nodes for each cargo are set at minimum and maximum in the $LP_2$ setting and at minimum and medium in the upper bound version. This means that the load quantity is smaller in half of the nodes. This way the number of feasible sequences and hence the solution space is larger in the upper bound version. This might explain the increase in CPU times. A reasonable upper bound can also be found by stopping the upper bound version after the first node. Table 4 shows that in some cases the bound is almost identical to the final bound, but for instance in Case 1 the optimal solution is 6.1% below the first node bound and only 1.5% below the final bound. The CPU gain from stopping after the first node also varies among the cases. Since checking the upper bound gap after the first node is costless in terms of computational effort, this check should be made, and if the gap is small enough the branch-and-price search could be stopped.

In a practical planning situation, the $LP_1$ setting should be used when the CPU time is critical, since it gives high quality solutions in short time. Otherwise, $LP_2$ should be used followed by the upper bound discretization so that we get near optimal solutions with a good upper bound.

Table 5 shows the results when the label domination criteria were approximated for the discretization approach (Section 3.5). The approximation was tested for the $LP_2$ setting. If the results are compared with Table 4, it can be seen that the approximation gave a decrease in computation time for all cases except cases 3 and 8. There was no change in the objective values.

<table>
<thead>
<tr>
<th>Cases</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gap from best solution</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B&amp;B nodes</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>184</td>
<td>4</td>
<td>4</td>
<td>42</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>CPU</td>
<td>13</td>
<td>7</td>
<td>10</td>
<td>2715</td>
<td>2047</td>
<td>21</td>
<td>7</td>
<td>215</td>
<td>592</td>
<td>215</td>
</tr>
</tbody>
</table>

5 CONCLUDING REMARKS

We have presented a column generation procedure for the tramp ship scheduling problem with flexible cargo sizes. A modified version of the dynamic programming method of Dumas et al.
(1991) is used to find the \( p \) best solutions of the subproblem with discretized load quantities. For all the \( p \) best subproblem solutions we optimize load quantities by using the LP model and solution algorithm introduced by Brønmo et al. (2006). If any of the columns have a positive reduced cost, it is transferred to the restricted master problem. The column generation scheme stops when no more positive reduce cost columns can be found. Finally we use a branch-and-price approach to find an integer solution.

Since the result from our column generation approach is not necessarily optimal, we calculate an upper bound to the problem. We introduce an upper bound discretization setup where the income of each of the fixed cargo sizes in the discretization set is increased, while the resource consumption is the same as in the original problem. The income is set so that any original problem solution corresponds to an upper bound discretization solution with equal or higher profit. Now, we find the optimal solution to the upper bound discretization setup by using the column generation scheme without quantity optimization, and we have computed an upper bound to the original problem.

In the computational study we have compared our method with the a priori column generation approach of Brønmo et al. (2006) on six cases based on real data. The results show that column generation is more efficient than a priori column generation also in the case of flexible cargo sizes. Our column generation approach found the optimal solution to all the six cases. In order to show that a more extensive set of practical instances can be solved, we have also presented four cases that were solved by our method but could not be solved by the a priori column generation algorithm. In these four cases the column generation solution was 0.7% – 3.0% from the computed upper bound. The upper bound gap can be divided into two parts; one part comes from the overestimation of the computed upper bound, and the other part is the real optimality gap. Since the computed upper bound overestimated the solution of the six benchmarked cases by 0.04% – 1.5%, we find it reasonable to claim that the real optimality gap of the four larger cases is smaller than the upper bound gap of 0.7% – 3.0%.

Our results indicate that in the case of relatively small or tightly constrained instances, the a priori column generation approach should be chosen since optimality is guaranteed. In larger or more loosely constrained cases the column generation approach gives a very good compromise between solution time and quality.

6 ACKNOWLEDGEMENTS

This work was carried out with financial support from the TOP project on Transport Optimization through the Research Council of Norway. The authors are grateful to Aarhus School of Business for hosting Geir Brønmo during this work. Geir Brønmo’s stay at Aarhus School of Business has been supported by a mobility scholarship from the Nordic Research Board (NordForsk).
REFERENCES


Christiansen M, Fagerholt K. Robust ship scheduling with multiple time windows. Naval Research Logistics 2002; 49(6); 611-625.


Ioachim I, Desrosiers J, Dumas Y, Solomon MM, Villeneuve D. A request clustering algorithm for door-to-door handicapped transportation. Transportation Science 1995; 29(1); 63-78.


Sigurd M, Pisinger D, Sig M. Scheduling transportation of live animals to avoid the spread of diseases. Transportation Science 2004; 38(2); 197-209.


Xu H, Chen ZL, Rajagopal S, Arunapuram S. Solving a practical pickup and delivery problem. Transportation Science 2003; 37(3); 347-364.
L-2006-07  Geir Brønmo, Bjørn Nygreen & Jens Lysgaard: Column generation approaches to ship scheduling with flexible cargo sizes.


L-2006-03  Christian Larsen: Computation of order and volume fill rates for a base stock inventory control system with heterogeneous demand to investigate which customer class gets the best service.

L-2006-02  Søren Glud Johansen & Anders Thorstenson: Note: Optimal base-stock policy for the inventory system with periodic review, backorders and sequential lead times.

L-2006-01  Christian Larsen & Anders Thorstenson: A comparison between the order and the volume fill rates for a base-stock inventory control system under a compound renewal demand process.

L-2005-02  Michael M. Sørensen: Polyhedral computations for the simple graph partitioning problem.


L-2004-03  Søren Glud Johansen & Anders Thorstenson: The (r,q) policy for the lost-sales inventory system when more than one order may be outstanding.

L-2004-02  Erland Hejn Nielsen: Streams of events and performance of queuing systems: The basic anatomy of arrival/departure processes, when focus is set on autocorrelation.

L-2004-01  Jens Lysgaard: Reachability cuts for the vehicle routing problem with time windows.