Chapter 8

SHIP SCHEDULING WITH RECURRING VISITS AND VISIT SEPARATION REQUIREMENTS

Mikkel M. Sigurd
Nina L. Ulstein
Bjørn Nygreen
David M. Ryan

Abstract This chapter discusses an application of advanced planning support in designing a sea-transport system. The system is designed for Norwegian companies who depend on sea-transport between Norway and Central Europe. They want to achieve faster and more frequent transport by combining tonnage. This requires the possible construction of up to 15 new ships with potential investments of approximately 150 mill US dollars. The problem is a variant of the general pickup and delivery problem with multiple time windows. In addition, it includes requirements for recurring visits, separation between visits and limits on transport lead-time. It is solved by a heuristic branch-and-price algorithm.

1. Introduction

Increased pressure on road networks and increasing transport requirements make companies look for new transport solutions. This spurred an initiative to create a new liner shipping service. The initiative came from a group of Norwegian companies who need transport between locations on the Norwegian coastline and between Norway and The European Union. While few producers on the Norwegian coast have sufficient load to support a cost efficient, high frequency sea-transport service, they can reduce costs and decrease transport lead-time by combining their loads on common ships. They agreed upon a tender (transport offer) which was proposed to a number of shipping companies. The tender specifies the number of cargos per week and time constraints for pickup

and delivery. It also states the requirements regarding ship-types and loading and unloading techniques. For rapid handling, all goods must be transported in containers. Finally the tender specifies the yearly payment each company will make to be part of this transportation system. Today there are neither ships nor harbour facilities to support the proposed solution. Thus, major investments are necessary. Estimates indicate that investments in ships alone, can amount to about 150 mill US dollars. We present a model which calculates an optimal solution to the requirements in the tender. The model includes selection of an optimal fleet composition, ship routing and visit-schedules. The problem is formulated as a set partitioning model and solved by a heuristic branch-and-price algorithm. The next section presents the system requirements in more detail. In Section 3 the problem is compared to other fleet design and routing problems. Our choice of master and pricing problem is presented in Sections 4 and 5. The branching strategy is described in Section 6. Section 7 presents results, while Section 8 conclude with some remarks on the model choice and on the results.

2. Problem description

In this section we will first take a closer look at the ship requirements, and then describe requirements pertaining to customers transport demand.

To achieve fast transport, it is necessary to limit both the travel time and the loading and unloading time. Faster ships can substantially decrease the travel time. While traditional cargo ships travel at about 16 knots, cargo ships can be designed to travel at up to 25 knots. With this speed a ship can travel from Trondheim to Rotterdam in 35 hours. This represents a reduction in travel time of about 20 hours compared to traditional cargo ships. Although higher speed increases variable costs, as fuel consumption for ships increase exponentially with speed, this may be outweighed by a reduction in the number of required ships, reduced inventory costs and the need to satisfy customers' lead-time requirements. Combining tonnage leads to an increased number of port visits. To limit the loading and unloading time, ships need to use a roll-on roll-off technology. This means that cargo is rolled onto the ships by trucks and not lifted by cranes. The existing fleet of ships serving the North-Sea region cannot adopt this technology. Therefore the system requires construction of new ships. The shipping companies have in collaboration with the customers proposed a number of candidate ship-types. It is possible to construct any number of each candidate ship-type. The candidate ships vary in cost, capacity and speed. Some ships have properties which

Table 8.1. Alternative visit-patterns for a customer with three visits per week and at least one day in-between visits.

nr	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*	*Sat*	*Sun*
1	X	O	X	O	X	O	O
2	X	O	X	O	O	X	O
3	X	O	O	X	O	X	O
4	O	X	O	X	O	X	O
5	O	X	O	X	O	O	X
6	O	X	O	O	X	O	X
7	O	O	X	O	X	O	X

prevent them from visiting particular harbours. The constructed ships will be used in full by the system. The fixed weekly cost of a ship covers crew costs, financial costs and maintenance costs. The financial cost of a ship equals the depreciation cost from constructing the ship. The variable cost depends mainly on the fuel consumption, which is calculated as a function of the travel distance and speed.

The tender includes transport of 68 cargos per week between 21 harbours, 20 in Norway and one in Rotterdam. The total transport volume is approximately 2000 containers weekly. All customers specify a pickup port and a delivery port, a weekly load and a frequency. The frequency states the number of shipments per week. The weekly load is distributed evenly among the shipments. For each shipment, there can be single or multiple *time-windows* for pickup and for delivery. If, for example a cargo can be collected between Monday and Wednesday but only within the opening hours of the port, there will be three time windows, one for each day. The maximum *lead-time* from pickup to delivery limits the time from when a cargo is picked up until it is delivered. Lead-time requirements apply to perishable goods such as fish and to goods where customers require rapid delivery. Customers with multiple visits per week, can demand a minimum time between visits or limit the number of visits during a given number of days. If a customer requires at least one day between visits in the pickup port and a ship visits on Monday, then visits on Sunday and Tuesday are forbidden. Table 8.1 shows the seven feasible *visit-patterns* for a customer with three cargos and at least one day in-between service. If the customer instead requests not more than two visits per three days, there are 21 additional feasible visit-patterns. It is possible to enumerate all feasible visit-patterns for customers with separation requirements for the visits. If desired, this

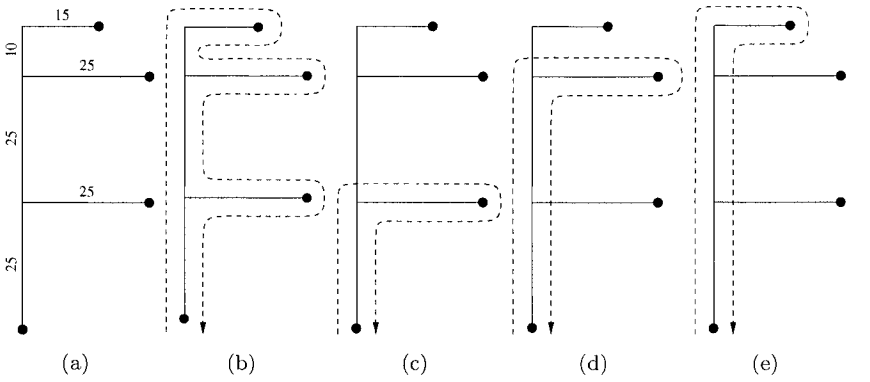

Figure 8.1. An example showing how extending the planning period from one week to two weeks can improve the solution.

can also be done for customers without requirements for separation of visits.

To facilitate planning, companies want a weekly recurring *visit-schedule*, similar to a bus-schedule. The visit-schedule must state the day when each visit is made, but does not restrict the time of day for making the visit. This implies that the same visit-pattern will be repeated each week for each customer. The visit-schedule is not given a priori, so the shipping companies must decide on an optimal visit-schedule which complies with the customers requirements. Let a route denote a sequence of pickup and delivery visits on given days made by a particular ship-type. A recurring visit-schedule can be met by a set of weekly recurring routes, as cyclic routes with the duration of one week will visit the same customer(s) on the same day each week. Alternatively, ships can use cyclic routes with a longer duration than one week. Then the requirement for weekly recurring visit-patterns at each customer must be fulfilled by combining routes. For example, if the route length is two weeks, two ships can alternate on visiting the same customers every second week. Figure 8.1 illustrates how it is possible to reduce costs by 50% by allowing two-week routes instead of one-week routes. Figure 8.1(a) depicts a problem instance with four harbours. The most southerly harbour is Rotterdam and the other three harbours are on the Norwegian coastline. The three Norwegian harbours must be visited once a week to pick up cargo bound for Rotterdam. The numbers on the edges indicates the length of the edge in hours of sailing. Thus the route shown in 8.1(b) takes 250 hours $\approx$ 10.5 days and it covers all visits. Thus two ships sailing this route, starting one week apart will cover all visits in every week. On the other hand, if the maximum route length is set to one-week, the

three routes shown in Figure 8.1(c)–8.1(e) are required to cover all visits, which means using an additional ship.

The duration of the routes will be a multiple of the length of the recurring visit-schedule, which is one week. Ship owners suggest a sensible duration for routes should be two weeks. This is therefore used in the further description. With a two week planning period, the planning period includes two weekly visit-schedules. After two weeks, the plan is repeated. The ships travel on two week cyclic routes and visit the depot at least once during this time. The ships can be anywhere on their route at the start of the planning period.
Based on the tender and on the candidate ship-types, the mathematical model produces:

- a fleet of ships,
- a published and fixed visit-schedule,
- a recurring route for each ship.

The model takes into account requirements for:

- separation of visits to the same customer,
- time-windows (multiple) for pickup and delivery,
- visits on same days each week,
- lead-time from pickup to delivery.

The properties of the routes are:

- start and end at the same harbour,
- maximum route length (in weeks),
- time for each visit,
- load always less than ship capacity,
- lead-time from pickup to delivery must be met,
- time window constraints must be met,
- port/ship compatibility must be met.

Unlike bus scheduling, the ships do not return back to the "garage" at regular intervals. However, one harbour is special and will be used as a depot. The southernmost harbour, Rotterdam, receives and sends large quantities of goods from and to Norwegian harbours. Since Rotterdam

is relatively far away from Norway, we can safely assume that optimal routes never bring goods destined for Norway, to Rotterdam and back to Norway. Thus, after unloading and before reloading, ships will be empty in this harbour. Because of the large portion of cargo destined for Rotterdam, all ships will travel from Norway to Rotterdam at least once during a planning period. Hence, Rotterdam is used as the depot for all ships. Ships can leave from the depot at any time during the week. The ships are allowed to visit the depot again within the route, but they need to return to the depot and discharge all cargo within the maximum route duration. It is possible to include additional depots, in particular, if the assumption that ships will always be empty in the depot still holds.

The model assumes deterministic transport demand. This is based on data from each of the collaborating companies. The sensitivity of the solution to changes in demand is further discussed in Section 8.

3. Similar problems in the literature

3.1 Sea-transport

Ship scheduling and fleet planning often involves decisions which represent large monetary values. Constructing or acquiring a ship costs millions of US dollars and daily operation costs amounts to thousands of dollars. Improved utilization and fleet planning can lead to great benefits. This should motivate the use of decision support. According to Ronen (1993), optimization based decision support was not often applied in the shipping industry before 1993. This lack of interest, (Ronen, 1983), explains as a result of strong traditions for other planning methods as well as a range of operational factors in which sea transport problems differs from vehicle routing problems. In a more recent review Christiansen, Fagerholt and Ronen (2004) report on an increase in the number of studies for maritime transport planning. As most of the studies consider industrial and tramp shipping, the literature on liner shipping problems is still sparse. This does not reflect the development in global capacity in liner shipping, which was nearly doubled from 1991 to 1995.

Most liner shipping problems consider fleet size and mix in addition to fleet deployment. Cho and Perakis (1996) present models for routing and fleet design in a liner shipping problem. They present a LP model for a problem with a fixed fleet of ships and a set of candidate routes. This becomes an IP problem when the number of ships is allowed to vary. The IP problem minimizes the costs of satisfying customers demand by assigning ships to routes. The candidate routes are suggested by the

planners. In a later application Powell and Perakis (1997) expand on the issue of fleet deployment and include penalties for days when ships are idle. Fagerholt (1999) describes another model for a liner shipping system. Because of limited total route duration, all routes are generated a priori. Then, from a set of predefined alternative ships, the least costly ship is assigned to each route and set partitioning is used to select among the routes. The limitation to all three approaches is that they only work on relatively constrained problems. In contrast to the former studies, Rana and Vickson (1991) present a profit maximization model which constructs routes of favourable visits and selects routes from this set. They apply a Lagrange decomposition approach to solve the problem. By relaxing the demand constraints, the problem is decomposed into one problem for each ship.

Christiansen (1999) describes another decomposition approach to solve an industrial shipping problem with transport of only one bulk product, but it also involves managing inventory held at the ports. Therefore, it includes decisions on both load quantity and routes. The solution approach uses Dantzig-Wolfe decomposition. Suggestions for routes and load quantities are generated in a subproblem. The master problem selects routes that minimize cost while controlling inventory levels. This problem is solved by a heuristic branch-and-bound algorithm with generation of new columns when needed.

3.2 Related problems

Similar problems to the liner shipping problem are found in freight transport, train scheduling and in the airline industry. Such problems are often referred to as service network design problems, see Crainic and Laporte (1997).

The demand requirements with sets of legal visit-patterns at each customer are similar to those given in periodic VRP (PVRP) problems. Cordeau, Gendreau and Laporte (1997) present the currently known best heuristic for the PVRP. However, as their method uses the fact that all routes in the PVRP last for only one day and also do not have pickup and delivery, their method is not directly relevant to our problem.

4. Mathematical model formulation

The model constructs a fleet of ships, a visit-schedule and routes. A Dantzig-Wolfe decomposition is applied to formulate a set partitioning problem which is in turn solved by a branch-and-price algorithm. Desrosiers et al. (1995) reports that this approach has been successfully applied to numerous routing and scheduling problems while Barnhart et

al. (1998) discuss how this method can be used to solve various classes of integer models.

The decomposition gives a *master problem* and a *pricing problem*. The master problem selects, from a set of candidate routes, routes which minimize the cost of satisfying customers requirements. With a huge number of possible routes only a subset of routes can be included in the master problem. New routes are generated in the pricing problem and added to the master problem. Dual values from the master problem are used to modify the objective function of the pricing problem to encourage new routes for poorly serviced cargoes. With delayed column generation, new routes are generated throughout the branching process. When no improving routes can be found the algorithm terminates. As further described in Section 5, heuristic methods are used to solve the pricing problem. This gives a heuristic branch-and-price algorithm.

The system must service a given number of cargos. Each cargo has an origin and a destination node. There may be time window constraints on pickup and delivery and possibly lead-time constraints for delivery of the cargo. Since new ships will be constructed, ship characteristics are not fixed. In theory, both speed and capacity of the ships could be modelled as continuous variables. However, shipping companies prefer some standardization. Therefore speed and capacity are modelled as stepwise functions. This results in a finite number of ship-types. There is no limit on the number of ships of each type.

4.1 The master problem

The master problem selects routes from a subset of all routes. Recall that a route is a sequence of pickup and delivery visits on given days made by a particular ship-type. There are two ways to model the composition of routes for the planning problem. One approach is to construct routes which last for the duration of the planning period in the pricing problem. This way, only complete routes are selected in the master problem and selecting a route also involves using a ship. Alternatively, a collection of shorter routes can be proposed to the master problem. Then a sequence of shorter routes for each ship are selected in the master problem. This requires additional constraints on the daily utilization of unique ships. Both approaches have been tested. The approach with complete routes gave better results. Therefore only this approach is further described.

The plan for one planning period is repeated at the end of the period. With complete routes which last for the duration of the planning period, a ship travels exactly one route. A ship can be anywhere on its route

at the beginning of the planning period. In other words, a route which start from the depot on day d in the planning period will wrap around and finish on day $d-1$ in the planning period. Let $\mathcal{S}$ denote the set of ship-types s, and $\mathcal{R}^s$ denote the set of candidate routes r for ship-type s. With complete routes, the cost of using a ship can be included in the cost for the route, C_r^s, and calculated in the pricing problem.

Let the variable z_r^s be one if route r, with ship-type s, is used. There are no restrictions on how many times each ship-type can be used, so different routes which use the same ship-type can be selected simultaneously. However, a particular route r can only be used once, as each visit can only be made once, so $z_r^s \in \{0,1\}$. The master problem selects routes z_r^s to minimize system costs, while ensuring that all requests are served. The cost minimization objective is given in (8.1).

$$\min \sum_{s \in \mathcal{S}} \sum_{r \in R^s} C_r^s z_r^s. \tag{8.1}$$

Each customer requires one or more shipments per week between its pickup port and its delivery port. The cargos which belong to one customer are identical in terms of load, lead-time requirements, origin and destination. Let $\mathcal{G}$ denote the set of all ports, and g denote a particular pickup or delivery port. Note that ports are customer-specific so that different ports may correspond to the same geographical location. The term "harbour" refers to a particular geographical location. A visit-pattern gives the legal days d of visit at port g in compliance with the separation requirements for the port. Each port g must be visited according to one of its feasible visit-patterns. Since customers want weekly recurring visits a one-week visit-pattern is repeated for each week in the planning period. With a one week recurring visit-schedule and a two week planning period, each visit-pattern consist of 14 days. Let $\mathcal{D}$ denote the set of all days in the planning period. Within these 14 days, a weekly pattern is repeated two times. Let $\mathcal{P}^g$ denote the set of feasible patterns p for port g. The first week $d = \{1, \ldots, 7\}$ of a visit-pattern with visits on Tuesday, Thursday and Saturday, can be expressed by: $\{\overline{M}_{g1}^p, \ldots, \overline{M}_{g7}^p\} = [0\ 1\ 0\ 1\ 0\ 1\ 0]$. The variable u_g^p is one when pattern p is used for port g. B_{gdr}^s is one when route r with ship s visits port g on day d. Restriction (8.2) requires that if a route visits port g on day d, then a visit-pattern p with a feasible visit on day d must be selected.

$$\sum_{s \in \mathcal{S}} \sum_{r \in R^s} B_{gdr}^s z_r^s - \sum_{p \in \mathcal{P}^g} \overline{M}_{gd}^p u_g^p = 0 \quad \forall g \in \mathcal{G}, \qquad d \in \mathcal{D}. \tag{8.2}$$

These pattern matching constraints can be reformulated as partitioning constraints by replacing the service pattern $\overline{M}_{gd}^p$, by the "inverse".

The "inverse" representation is: $\{M^p_{g1}, \ldots, M^p_{g7}\} = [1\ 0\ 1\ 0\ 1\ 0\ 1]$. In addition, restriction (8.4) ensures that exactly one pattern is used for each customer. This requirement also ensures that all cargos are served exactly once. The master problem can now be expressed as:

$$\min \sum_{s \in \mathcal{S}} \sum_{r \in R^s} C^s_r z^s_r \tag{8.3}$$

subject to

$$\sum_{p \in \mathcal{P}^g} u^p_g = 1 \quad \forall g \in \mathcal{G}, \tag{8.4}$$

$$\sum_{s \in \mathcal{S}} \sum_{r \in R^s} B^s_{gdr} z^s_r + \sum_{p \in \mathcal{P}^g} M^p_{gd} u^p_g = 1 \quad \forall g \in \mathcal{G}, \qquad d \in \mathcal{D}, \tag{8.5}$$

$$z^s_r \in \{0,1\} \quad \forall s \in \mathcal{S}, \quad r \in R^s, \tag{8.6}$$

$$u^p_g \in \{0,1\} \quad \forall g \in \mathcal{G}, \quad p \in \mathcal{P}. \tag{8.7}$$

The combination of routes selected in the master problem must visit each port on days which comply with a feasible visit-pattern for the port. Selecting a visit-pattern fulfills both the requirement for weekly recurring visits and for separation of visits. In addition, it ensures that all cargos from/to the port are collected or delivered. Since pickup and delivery restrictions are fulfilled in all routes which are proposed by the pricing problem, it is sufficient with a pattern restriction in either the pickup port or the delivery port for each customer to ensure that all shipments are made. In this system the customers want both regular departures and arrivals of goods and therefore most customers specify separation requests in both pickup and delivery ports.

Constraints (8.4) and (8.5) have dual variables π_g and δ_{gd} respectively. Since constraint (8.4) does not involve the route variable z^s_r, only the δ_{gd} duals influence costs in the pricing problem.

5. The pricing problem

When solving the master problem using delayed column generation, columns must be added during the branching process. Finding or *generating* suitable columns to add to the master problem is known as the pricing problem. Only columns with negative reduced costs are added to the master problem, since only these may go into the basis.

The reduced cost of a route equals the violation of the corresponding constraint in the dual linear program. Thus, the reduced cost $\bar{c}^{\pi,\delta}_{s,r}$ of route $r \in R^s$ is:

$$\bar{c}^{\pi,\delta}_{s,r} = C^s_r - \sum_{g \in \mathcal{G}} \sum_{d \in \mathcal{D}} B^s_{gdr} \delta_{gd}.$$

Adding a negative reduced cost column corresponds to adding a violated inequality in the dual linear program. If no negative reduced cost column exists, the optimal solution of the restricted master problem is also optimal for the complete master problem. The pricing problem consists of finding one or more legal routes with negative reduced cost.

The pricing problem is represented in a directed graph with edge weights. For each pickup (or delivery) port g and for each day d, create a node i, where $i = f(g, d)$. Furthermore, add a depot node 0 and add edges from the depot node to all pickup nodes and from all delivery nodes to the depot node. Also add edges between all feasible node pairs. A node pair (i, j) is feasible if there is sufficient time to visit i before visiting j given the time windows of i and j. The cost C_r^s of a route r sailed by ship s, consists of a fixed cost and a sailing cost. The fixed cost is independent of how we use the ship. The sailing cost, however, is proportional to the distance traveled by the ship. Both costs depend on the ship-type used.

The sailing cost can be assigned to edges going into the visits. Given a dual vector $\bar{\delta}$, the value of dual δ_{gd} can be subtracted from the edge going into node $i = f(g, d)$. In this graph, the reduced cost of a route equals the length of the path in the graph consisting of a sequence of nodes corresponding to the visits on the route in that order.

To allow routes to wrap around the to the beginning of the planning period, append a copy of the network, less the depot node, to the end of the horizon. Add arcs from all delivery nodes in this network to the original depot node. Then solve one subproblem for each ship-type and for each day in the planning period.

Since the reduced cost of a route is equal to the length of the path in the above graph, the pricing problem can be solved as a shortest path problem with additional constraints that disqualify illegal routes. This problem is commonly known as a *resource constrained shortest path problem* (RCSP). In our case the additional constraints will make sure that all routes comply with the capacity, pickup and delivery, time-windows, lead-time and visit-pattern requirements which are described in Section 2.

Several methods have been proposed to solve the resource constrained shortest path problem, see Mehlhorn and Ziegelmann (2000). However, when dealing with non-additive "difficult" constraints like "pickup before delivery", "time windows", and "visit-patterns" only dynamic programming algorithms have been used to solve the problem. Dumas, Desrosiers and Soumis (1991) were the first to propose a dynamic programming algorithm for the pickup and delivery problem with time windows used in a column generation setting like this.

We solve the resource constrained shortest path problem with a heuristic two Phase algorithm. First, we use the fact that most cargos go between Rotterdam and ports on the Norwegian coastline. Because of this, we can assume that all ships will visit Rotterdam at least once in a planning period, either to pickup cargo, to deliver cargo or both. As explained in Section 2, since Rotterdam is relatively far away from the Norwegian ports we can assume that a ship unloads all cargo whenever it visits Rotterdam and does not carry cargo between two Norwegian ports through Rotterdam.

Following our assumptions, a route is comprised by a number of tours starting and ending with an empty ship in Rotterdam. Phase I of the algorithm will generate these tours and Phase II will collect the tours into complete routes. The fixed cost of using a ship of type s, is added in Phase II. A heuristic dynamic programming algorithm is used to solve Phase I and a k-shortest path algorithm with a feasibility check is used to solve Phase II. The two phases are described in detail below.

5.1 Creating tours

Creating good legal tours is the main part of the pricing problem. With a planning period of two weeks the problem involves 272 visits on 10 different ship-types. This is more than similar algorithms have been able to solve. Running an exact dynamic programming algorithm on problem instances of this size is too time-consuming because of the combinatorial complexity. A heuristic dynamic programming algorithm is designed in order to make the problem practically solvable.

The basic idea of the algorithm is similar to previously proposed dynamic programming algorithms for solving resource constrained shortest path problems, see Dumas, Desrosiers and Soumis (1991). We start with an empty ship in Rotterdam and extend the route to each of the ports. For every visit, we create a new label which contains all relevant information on the route so far. We check that candidate visits do not make the route illegal, i.e. we check that:

- We do not deliver a cargo that has not yet been picked up.
- We do not visit Rotterdam with any cargos bound for other ports.
- The ship has available capacity for picking up the cargo.
- The visit is made within an open time window. If there is a later time window, we wait.
- The visit is consistent with a visit-pattern for this node, especially if the same visit has been made previously on this route.

- We still have time to deliver all onboard cargos within their lead-time and within the planning period. (This corresponds to solving a travelling salesman problem with time windows by complete enumeration for the earliest cargos).

If any of the constraints are violated, the route is not extended to make the visit. As stated, the number of states created in the dynamic programming algorithm has to be pruned further in order to run the algorithm in practice. Thus, a number of heuristic constraints that the routes must satisfy are added. For each visit we check that:

- The waiting time is no more than 12 hours. We enforce this rule since good routes will not have long periods of waiting in ports.
- All cargos on the ship bound for a particular harbour are delivered when a ship visits the harbour. We enforce this rule since we expect that good routes will unload the cargos as soon as possible to make room for picking up more cargo.
- The route should not change direction more than five times on the Norwegian coast line. We enforce this rule since we expect that good routes will not make too many detours.
- All deliveries can be made without changing directions, if the route has already changed direction five times.

These additional constraints help prune away unpromising routes while running the dynamic programming algorithm. However, many routes will comply with these constraints, so additional pruning is needed. For this purpose the algorithm is constrained to a *limited subsequence* of visits for every state we create. For a given state, we list all possible visits we can make next. Each visit is assigned a score based on the route so far and on the best continuation of the route. Instead of creating new states for all possible next visits, we pick the three best scoring pickups and the three best scoring deliveries and create states only for those. The score equals the reduced cost of the visit in question plus the minimum reduced cost of delivering the cargos on board after this visit. Thus, the best subsequences of a given visit may change in every pricing iteration since the dual variables have changed. Also, the best subsequences of a visit may be different depending on which day we make the visit, since we have a different dual variable for every visit for every day of the planning period.

We use *dominance* to further reduce the number of dynamic programming states. If two states are placed on the same node and have made the same number of pickup and deliveries and one state has arrived there

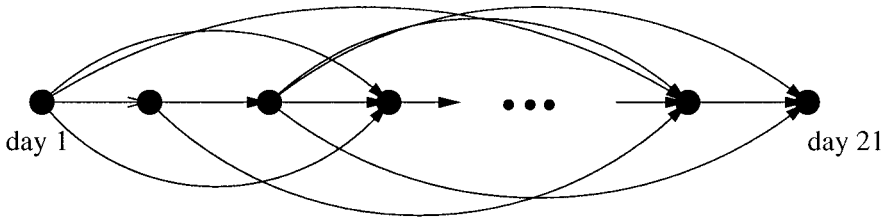

Figure 8.2. An auxiliary graph representing tours.

before the other with a lower reduced cost and one has been at least as far north as the second state, the second state is removed. This is done to remove unpromising routes that are similar, but inferior, to other routes at an early stage. Whenever the ship visits the depot, it has created a complete, legal tour. If the tour has negative reduced cost, it is saved in our pool of tours. The tours are not added directly as columns in the master problem, since they are typically much shorter than the planning period. Instead, tours from the tour pool are combined to create full length routes, which are added to the master problem.

5.2 Combining tours

Given a pool of tours created by our dynamic programming algorithm described above, the pricing problem is reduced to combining one or more tours into legal, full length routes. This is done by running a k-shortest path algorithm on an auxiliary graph, which is defined as follows:

- For every day in the planning period, we add a node in the auxiliary graph. The nodes constitute a time-line.
- For every node, we add an edge from the node to the next node on the time line. The edge weights are 0 on these edges. We do not add an edge going out of the last node.
- For every tour found in Phase 1, we add an edge from the node corresponding to the start time to the node corresponding to the end time of the tour. The edge weights are calculated in every column generation iteration as the reduced cost of the tours.

The graph layout is shown in Figure 8.2. The weights on the edges representing the tours equal the reduced cost of the tours. Minimum reduced cost combinations of tours can be found by a shortest path algorithm on the graph in Figure 8.2 with node i as source and node $i + 21$ as target. To handle tours that cross the end of the planning

period (e.g. a tour that starts on day 18 and ends on day 4), another 21 nodes are added to represent day 1 to 21 in the next planning period.

Combining tours with conflicting visit-patterns gives infeasible routes. Conflicting visit-patterns occur if tours visit the same customer on days which do not match any legal visit-pattern. A constrained shortest path algorithm can be used to find a legal minimum cost combination of tours. Since we wish to return more than one combination in every pricing iteration, a constrained k-shortest path algorithm can be used to solve the pricing problem.

Traditionally, this problem has been solved by a dynamic programming algorithm, where states are expanded by combining tours, whilst checking that the combination of tours are legal with respect to the visit-patterns. However, checking whether two tours have contradicting visit-patterns is rather time consuming. Furthermore, most tour combinations will be legal. For these reasons, the time-consuming check is not performed in the k-shortest path algorithm. Instead we first run an unconstrained k-shortest path algorithm as proposed by Carlyle and Wood (2003) to find a number of shortest paths which may or may not be legal. This algorithm first identifies the shortest path. Then it returns all paths which are within a factor of $1+\varepsilon$ of the shortest path for some specified ε. This is done by enumerating subpaths while checking if candidate subpaths can be extended to paths of acceptable length. Afterwards, a rather expensive visit-pattern check examines the routes, starting with the cheapest route and continuing until the desired number of legal routes has been found. Whenever an illegal combination of tours appears, this combination is inserted into a data structure which allows fast lookups. If the same tour combination appears in a later pricing iteration, it can be discarded without running the expensive check. In fact, the fast lookup can also be done in the k-shortest path algorithm, to avoid combinations, which have already been found illegal. A tree data structure with a sorted sequence of tour-numbers as keys is used to provide the fast lookups.

The running time of the k-shortest path algorithm depends on the number of paths to return. Therefore, only a small number of paths are returned early in the convergence. If there are too few legal paths within these paths, the number of paths to return is increased and the algorithm is re-run. In practice, the k-shortest path algorithm by Carlyle and Wood (2003), combined with a post-check and a fast lookup data structure performs very efficiently.

Both the Phase I algorithm which creates tours and the Phase II algorithm which combines the tours are run for every ship-type and every day in every pricing iteration. The fixed cost of using a particular ship-

type is added to the cost of each route in Phase II. Once a tour has been added to the tour pool, it is never removed. Hence, all generated tours are available for the tour combining algorithm in later pricing iterations.

6. Finding integer solutions

The optimal solutions to the linear relaxation of the master problem will generally not be integer. This means that in the optimal solution to the relaxed problem, more than one route variable at fractional value cover the transport of some cargo. Constraints (8.5) which include route and pattern variables may also permit the transport of some cargo by pattern variables at fractional values. The pattern variables will be integer in solutions where all the route variables are integer. If all pattern variables are integer, the resulting problem is a general pickup and delivery problem with time-windows. While it is hard to rank patterns, problem knowledge can be used to identify desirable routes. Therefore, the branching scheme works on the route variables, driving them towards integer values. This will simultaneously give integer pattern variables.

6.1 Branching on pairs of visits

We apply the branching strategy proposed by Ryan and Foster (1981) which has proven successful in a number of set partitioning applications. Recall that a row in the constraint matrix correspond to a particular visit on a given day. Following the Ryan–Foster branching strategy, it is possible to choose two rows k and l in the matrix such that the sum of route variables where the corresponding visits occur consecutively is strictly between 0 and 1. This corresponds to choosing two visits (n_1, d_1) and (n_2, d_2), where a visit is defined by a pickup (or delivery) n_i of a cargo group and a day d_j. Given two visits, on the first branch, demand that the pickup (or delivery) n_1 on day d_1 is followed by the pickup (or delivery) n_2 on day d_2. On the second branch, demand that the two visits are not made consecutively on those specific days.

The master problem changes on both branches, since we must remove variables which violate the branching restriction. On one branch we must remove all route variables which make one of the two visits n_1 on d_1 or n_2 on d_2, but not both. On the other branch we must remove all route variables which make both visits n_1 and n_2 consecutively on the specified days. The variables are not permanently removed, but rendered inactive, so that they can be easily put into the master problem again in other branches of the branch-and-bound tree. To prevent generating and adding violating variables to the master problem, the pricing problem needs to be changed. In any node in the branch-and-bound tree, the pric-

ing problem must reflect the restrictions of the branch. Phase II of the pricing algorithm works on a pool of tours, represented by edges in a directed acyclic graph. To avoid generating routes violating the branching constraints, all tours which violate the constraints are removed. Again, the tours are not really removed, but merely hidden so that they can be easily restored in other branches of the branch-and-bound tree. We do not branch on consecutive depot visits. All branching decisions are imposed within the tours which means that the branching decisions do not restrict the combination of tours which is done in Phase II of the pricing algorithm. However, it is necessary to prevent generating new tours which violate branching constraints. Hence, the dynamic programming algorithm, which constitutes Phase I of the pricing algorithm, checks if any branching constraints are violated when building the tours, discarding the violating tours.

6.2 Using problem information in branching

Although the above branching strategy will work for any choice of visits (n_1, d_1), (n_2, d_2), the performance of the branching strategy is much improved by selecting a sensible pair of consecutive visits. This is guided by knowledge of the problem and by information from the fractional solutions to the master problem. For every pair of rows in our constraint matrix (i.e. for every pair of days and every pair of pickup/deliveries) a score is calculated based on this information. In a depth first strategy we branch on the pair of rows with best score.

As mentioned, the score is based on knowledge of the problem instance and information from the fractional solution at hand. The information from the problem instance tells us whether two consecutive visits fit well together. For example, if the visits are far apart in either time or geography, the visits receive a low score, which implies that the visits do not fit well together. More specifically, for a visit in node n_1 on day d_1 followed by a visit in n_2 on day d_2 the following factors are used in the score calculation:

- A penalty proportional to the distance between n_1 and n_2.
- A penalty proportional to the time difference d_1–d_2 less the average travel time from n_1 to n_2.
- A penalty if the *direction* of cargos does not fit. For example, if n_1 is a pickup of a cargo bound for a harbour north of n_1 and n_2 lies south of n_1, the ship will need to go south to n_2 and then go north later on. Such detours are penalized. There are 14 cases similar to the above where the directions of two cargos do not fit

and where making the visits consecutively gives a detour. In all cases the penalty is proportional to the length of the detour.

The choice of visits to branch on should also be guided by the fractional solution at hand, since routes with high fractional values in an optimal LP-solution are likely to be good routes in an IP-solution as well. The sum of fractions is calculated for every day and every consecutive pair of visits. The sum of fractions for a particular visit pair is the sum of the values of the route variables that make both these visits on these days. The sum of fractions is added to the score, so that a high sum of fractions results in a higher score. This is based on the assumption that routes chosen in the optimal solution to the relaxed master problem are similar to the routes in an optimal solution to the IP master problem.

As stated, the pair of visits with the highest score is selected for branching. This pair constitutes visits which we think are likely to be made consecutively on a route. On the first branch these visits must be made consecutively on two specific days and on the other branch they can not be made consecutively. Since we pick pairs of visits which fit well together, we will follow the first branch most often, which is the stronger of the two since it fixes the two visits together.

7. Computational results

The heuristic branch-and-price algorithm described in Section 6 has been implemented and tested on test instances based on data provided by the collaborating companies. As could be expected, our results show that there are substantial savings in making two-week planning periods compared to one-week planning periods. The best solution found using a one-week planning period requires 15 ships and has an objective value of 20.28, whereas the best solution found using a two-week planning period only requires 13 ships and has an objective value of 17.67 per week. This constitutes a saving of 14.8%. In Table 8.2 we show some details of the performance of the branch-and-price algorithm on the two test instances.

The two week planning period is advantageous compared to the one week planning period because it allows a greater flexibility in the length of the tours. Where all the tours in the one week planning period are between 5 and 7 days long, the tours of the two week planning period are between 4 and 13 days long. Hence in the two week planning period each ship make either a single long tour, two medium tours or a short tour and a longer tour. In the one week planning period all ships make medium length tours.

	1 week instance	2 week instance
No. of constraints	624	1170
No. of columns	1624	10128
No. of B&B-nodes	45	107
No. of pricing itr.	124	565
CPU time (sec.)	440	40 576
Best solution found		
No. of ships	15	13
Cost pr. week	20.28	17.67

Table 8.2. The table shows details of running the branch-and-price heuristic on the test instances based on data provided by the collaborating companies.

8. Concluding remarks

We have chosen to solve the problem using column generation. Another approach would be to enumerate all possible combinations of feasible visit-patterns in all ports. This would give a finite number of alternative visit-schedules. For each visit-schedule one could solve a pickup and delivery problem with time windows and with lead-time as the only additional requirement. Unfortunately, the number of visit-schedules renders this approach impractical. The problem has 78 ports each with an average of 7.1 alternative patterns. This gives about $78^{7.1}$ (10^{13}) alternative visit-schedules or problems to solve. Various heuristic approaches could also be used. However, most would have difficulties incorporating the separation of visits, recurring visits and lead-time constraints. Since these restrictions require considering all routes simultaneously.

Demand is modelled as fixed. This is realistic, as the companies will have to pay for the requested capacity irrespective of whether or not they use it. Likewise the shipping company must guarantee a minimum transport capacity to each partner. It is likely that unused capacity will be traded among the partners and also sold to customers outside the system. This way there can be room for some variations in the load. The shipping companies need to consider whether they assume a net increase in total transport load and thus want to use ships with surplus capacity.

Acknowledgments Companies who have contributed to this work are Elkem, Norsk Hydro, Norske Skog and Statoil. The work has also involved the research institution MARINTEK.

References

Barnhart C., Johnson E.L., Nemhauser G.L., Savelsberg M.W.P., and Vance P.H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329.

Carlyle W.M. and Wood R.K. (2003). Near-Shortest and K-Shortest Simple Paths. *Technical Report*, Department of Operations Research, Naval Postgraduate School, Monterey, CA 93943, USA.

Cho S.-C. and Perakis A.N. (1996). Optimal liner fleet routing strategies. *Maritime Policy & Management*, 23:249–259.

Crainic, T.G. and Laporte G. (1997). Planning models for freight transportation. *European Journal of Operational Research*, 97:409–438.

Christiansen, M. (1999). Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33:3–16.

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

Cordeau, J.-F. Gendreau, M., and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In: *Network Routing* (M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, eds.), volume 8, Handbooks in Operations Research and Management Science, pp. 35–139, North-Holland, Amsterdam.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22.

Fagerholt, K. (1999). Optimal fleet design in a ship routing problem. *International Transactions in Operations Research*, 6:453–464.

Laporte, G. and Osman, I.H. (1995). Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262.

Mehlhorn, K. and Ziegelmann, M. (2000). Resource Constrained Shortest Paths. *Proc. 8th European Symposium on Algorithms* (ESA2000), pp. 326–337, LNCS 1879 Springer, Berlin.

Powell, B.J. and Perakis, A.N. (1997). Fleet deployment optimization for liner shipping: an integer programming model. *Maritime Policy & Management*, 24:183–192.

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

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

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

Ryan, D.M. and Foster, B.A. (1981). An integer programming approach to scheduling. In: *Computer Scheduling of Public Transport* (A. Wren, ed.), North-Holland Publishing Company.