Solving stochastic ship fleet routing problems with inventory management using branch and price

Solving Stochastic Ship Fleet Routing Problems with Inventory Management Using Branch and Price

Ken McKinnon and Yu Yu

Abstract This chapter describes a stochastic ship routing problem with inventory management. The problem involves finding a set of least cost routes for a fleet of ships transporting a single commodity when the demand for the commodity is uncertain. Storage at supply and consumption ports is limited and inventory levels are monitored in the model. Consumer demands are at a constant rate within each time period, and in the stochastic problem, the demand rate for a period is not known until the beginning of that period. The demand situation over the time periods is described by a scenario tree with corresponding probabilities. A decomposition formulation is given and it is solved using a Branch and Price framework. A master problem (set partitioning with extra inventory constraints) is built, and the subproblems, one for each ship, are solved by stochastic dynamic programming and yield the columns for the master problem. Each column corresponds to one possible tree of actions for one ship giving its schedule loading/unloading quantities for all demand scenarios. Computational results are given showing that medium sized problems can be solved successfully.

Keywords Stochastic Dynamic Programming • Branch and Price • Ship Routing • Inventory Management.

Introduction

The marine shipping industry has experienced an unprecedented boom over the past decade. This is not only because of the rapid growth of the requirements to transfer more and more energy and commercial commodities from one location to another, but also because the characteristics of the ocean shipping industry, with its low transportation costs and huge loading capacity, are suitable for cheaply transporting huge amounts of products.

K. McKinnon (✉) • Y. Yu
School of Mathematics, University of Edinburgh, Edinburgh, UK
e-mail: k.mckinnon@ed.ac.uk; yu.yu@pi-solution.com

© Springer International Publishing Switzerland 2016
P.M. Pardalos et al. (eds.), *Advances in Stochastic and Deterministic Global Optimization*, Springer Optimization and Its Applications 107,
DOI 10.1007/978-3-319-29975-4_8

The classical routing and scheduling problem for vehicles and ships is important part of the general transportation problem, and has received a great deal of attention in academic research. A large number of possible solution approaches have been presented in the literature, involving either exact optimization methods or heuristic algorithms. A comprehensive review is provided in [10]. This focuses on literature about ship routing and scheduling published between the years 1990 and 2003. The survey is presented in several different parts: strategy planning problem, tactical and operational planning problems, naval problems, and other related problems. A survey of different solution methods in the literature is also presented in the review. A mixed integer programming (MIP) model is described in [25] for the problem of transporting different bulk products from a set of origins to a set of destinations by a fleet of ships. A ship has separate compartments for different products. A ship's voyage goes from a single loading port to a single discharging port. A cost-based heuristic algorithm is also presented to obtain acceptable solution quickly. Sherali et al. [26] have presented an MIP model for the Kuwait Petroleum Corporation (KPC) problem. Because of the integrality conditions and the large number of demand contract scenarios, the problem cannot be solved to optimality by the MIP model. An alternative aggregated model is then formulated and solved by a specialized rolling horizon heuristic method to make the problem solvable. In the ocean shipping industry, expert opinion is an important factor. Crary et al. [11] introduce a model integrating the expert opinion and MIP model for the problem of sizing the US destroyer fleet. MIP models for SRP are also built in [3, 24, 27]. Heuristics are developed in [19] in order to obtain an acceptable solution within reasonable time when solving the MIP model.

The Dantzig–Wolfe decomposition approach has proved to be successful for the vehicle routing problem with time windows. Desrochers et al. [14] were the first to propose a set partitioning model for the vehicle routing problem with time windows solved by column generation, and this appears to be an efficient way of finding the optimal solution. As for the ship routing problem, it is also a good solution approach. There is much literature on solving the problem by Dantzig–Wolfe decomposition. Early papers [1, 2] describe a typical tramp ship scheduling problem, which were the first works to use a Dantzig–Wolfe decomposition approach for ship routing and scheduling. The master problem is the linear relaxation of a set partitioning problem and subproblems are shortest path problems. But the algorithm presented cannot guarantee optimal integer solutions. In [5, 8, 9], the demand is regarded as a constant and Branch and Price is used to solve the problem. The problem is decomposed into a ship route subproblem for each ship and a port inventory subproblem for each port. The approach presented in this chapter is closest to that used in their paper. Compared to their papers, the present chapter deals with different inventory situations, and solves the stochastic problem rather than deterministic problem.

In realistic shipping operations, especially for ocean shipping, much of the planning data is uncertain. Deterministic models for ship routing and scheduling are sometimes inappropriate, and there is a need to develop stochastic model.

The stochastic vehicle routing problem (SVRP) without inventory constraints and with simple recourse actions is discussed extensively in the literature. A Branch and Price algorithm for vehicle routing problem with stochastic demands is presented in [7]. In this paper, the expected number of failures and the corresponding penalty cost are considered in the objective function and a two stage stochastic program with fixed recourse and capacity constraints is built. A straight-forward modification of the Clark and Wright savings algorithm for the SVRP based on a discussion of route failure is presented in [16]. In [18, 20], an integer L-shaped method is used to solve SVRP to optimality. In [4] an a priori sequence among all customers of minimal expected total length is proposed, and a variety of theoretical approaches are analyzed as well. In addition, several solution frameworks for the stochastic vehicle routing with stochastic demands are discussed in [17].

There are few references in the literature to stochastic ship routing problems with inventory. A robust ship scheduling with multiple time windows is presented in [6]. This uses a set partitioning approach with the columns found a priori to minimize the chances that ships stay idle in ports during the non-working days. A Markov decision process model of the stochastic inventory routing problem is introduced in [23], and approximation methods are used to find acceptable solutions.

This chapter considers the problem of optimizing the distribution of a single commodity by a fleet of ships when there is limited storage at the supply and consumption ports and the consumer demand is uncertain. Consumer demand is described by a scenario tree and demand is assumed to be constant within each period. A solution consists of a tree of schedules for each ship, where a schedule for a ship specifies the loading and unloading quantities at each port visited and the start time of each such operation (which we refer to as a *service*). These ship schedules must be such that the storage limits at ports are satisfied at all times. The problem is formulated as a multistage stochastic programming problem and is solved by Branch and Price—a Branch and Bound method that uses Dantzig–Wolfe decomposition to solve each node. The master problem is a set partitioning problem with extra inventory constraints. Each column in the master problem corresponds to a tree of schedules for a ship. Attractive columns are generated by a stochastic dynamic programming using a backward labelling method.

The structure of the rest of the chapter is as follows. Section "Decomposition Approach for the Stochastic Ship Routing Problem" introduces the Dantzig–Wolfe decomposition approach and describes the structure of the master and subproblems. This section also describes the techniques used to eliminate cycles. Sections "Branch and Bound" gives the Branch and Bound algorithm and "Examples and Results" present computational results, and section "Conclusion" gives the conclusion.

Decomposition Approach for the Stochastic Ship Routing Problem

Assumptions

The ocean transportation problem is too complex to consider every factor in the real world when modelling the problem. In this chapter we make the following simplifying assumptions:

- At each consumer port, the rate of demand is constant within a period, but can change between periods.
- At each port, loading and unloading rates are constant.
- At most one ship can be loading or unloading at any given time. This assumption avoids the overlap of services at a port.
- For each ship the travel time and cost between any two ports are fixed.
- A service at a port must start and finish within a period. Note, however, this is not a practical limitation as the service can continue without a break in the following period.

Solution Framework

A Branch and Price algorithm is used in this chapter. This consists of a master problem which is solved by Branch and Bound (B&B), with each node in the B&B tree being solved by Dantzig–Wolfe (DW) decomposition. Each column in the master problem corresponds to a tree of schedules for a ship. There is a huge number of these columns and if all were included explicitly the master problem would be impossible to solve. However the DW approach only generates the small subset of them that are needed, and is thus able to solve the full problem at each B&B node. In each iteration of DW a subproblem is solved for each ship to generate an attractive tree of schedules for that ship. In this chapter the subproblems are solved by stochastic dynamic programming.

At any stage in the solution of a master problem at a B&B node, a (finite) subset of the columns will have been generated. This problem, called a restricted master problem, is solved and the shadow prices of the constraints are then used to find the most negative reduced cost from among the columns that have not yet been generated. This can be done without explicitly generating any columns by solving a stochastic dynamic programming problem separately for each ship. The solution gives the tree of schedules for the ship. If this added as a column to the master problem, it would have the most negative reduced cost among all the possible columns for that ship. This procedure continues until no column with negative reduced cost can be generated, at which stage the master problem for that B&B node has been solved.

Master Problem

The detail formulation of a master problem is introduced here. A port can be visited several times within the time window of a scenario tree node, so an index for visit number is needed. In the model, many objects are indexed by the triple (Port, Visit, Scenario node) which is referred as a port visit. For any ship, there are a set of trees of schedules for it. The problem is to choose one tree of schedules for each ship. We introduce the details of master problem as below.

Indices

i	port
k	scenario tree node
$a(k)$	predecessor node of node k in scenario tree
m	the order of the visit to a port within a scenario tree node (i.e., the current visit is the mth time the port has been visited in this node)
v	ship
s	tree of schedule for one ship
(i,m,k)	a port visit

Sets

N	set of ports
V	set of ships
K	set of scenario tree nodes
K^T	set of scenario tree nodes in final period
P	set of port visits
R_v	set of tree of schedules for ship v

Parameters

A_{svimk}	the number of times in tree of schedules s for ship v that it makes port visit (i,m,k) (must be 0 or 1 to be feasible)
C_{sv}	expected cost if ship v takes the tree of schedules s
Q_{svimk}	quantity unloaded by ship v in port visit (i,m,k) if ship makes that port visit in schedule tree s, and 0 otherwise (value is negative if ship is loading)
T_{svimk}	the start service time for ship v in (i,m,k) if the ship makes that port visit in schedule tree s, and 0 otherwise
B_k	end of the time period which includes scenario tree node k
W_i	unloading rate from ship to port i (value is negative if ship is loading)
M	the maximum number of visits to any port in a scenario tree node
D_{ik}	demand rate in port i in node k (value is negative at a supply port)
S_i	initial stock level in port i
$\bar{S}_i$	upper bound for storage in port i
$\underline{S}_i$	lower bound for storage in port i

The values of parameters A_{svimk}, Q_{svimk}, and T_{svimk} are found by solving subproblems. These three parameters represent the route information and all three are either zero or all three are nonzero. A_{svimk} is the number of times ship v makes port visit (i,m,k) in schedule s, and to be feasible this must either be 0 or 1. However during the solution process there may be cycles of port visits, and this leads to values of A_{svimk} greater than 1. Such infeasible schedules cannot be included in the final optimal solution. Parameters Q_{svimk} and T_{svimk} represent, respectively, the quantity loaded and the start service time for this port visit.

Variables

x_{sv}	1 if ship v takes schedule tree s, and 0 otherwise
y_{imk}	1 if some ship makes port visit (i,m,k), and 0 otherwise
q_{imk}	amount of commodity unloaded from a ship during port visit (i,m,k) (value is negative if ship is loading)
t^S_{imk}	the start of service time in port visit (i,m,k)
t^E_{imk}	the end of service time in port visit (i,m,k)
h^S_{imk}	the stock level at time t^S_{imk}
h^E_{imk}	the stock level at time t^E_{imk}

Formulation of Master Problem

$$\min \sum_{v\in V}\sum_{s\in R_v} C_{sv}x_{sv} \tag{1}$$

$$\sum_{v\in V}\sum_{s\in R_v} A_{svimk}x_{sv} = y_{imk} \quad \forall (i,m,k)\in P \tag{2}$$

$$\sum_{v\in V}\sum_{s\in R_v} Q_{svimk}x_{sv} = q_{imk} \quad \forall (i,m,k)\in P \tag{3}$$

$$\sum_{v\in V}\sum_{s\in R_v} T_{svimk}x_{sv} + (1-y_{imk})B_k = t^S_{imk} \quad \forall (i,m,k)\in P \tag{4}$$

$$\sum_{s\in R_v} x_{sv} = 1 \quad \forall v\in V \tag{5}$$

$$x_{sv} \geq 0 \quad \forall v\in V, s\in R_v \tag{6}$$

$$\{x_{sv} : s\in R_v\} \text{ yield a valid tree of schedules for ship } v,\ \forall v \tag{7}$$

$$y_{imk}\in\{0,1\} \quad \forall\ (i,m,k)\in P \tag{8}$$

$$t^E_{imk} = t^S_{imk} + q_{imk}/W_i \quad \forall (i,m,k)\in P \tag{9}$$

$$t^E_{i,m-1,k} \leq t^S_{imk} \quad \forall (i,m,k)\in P,\ m>1 \tag{10}$$

$$y_{imk} \geq y_{i,m+1,k} \quad \forall (i,m,k)\in P \tag{11}$$

$$h^E_{imk} = h^S_{imk} - (t^E_{imk} - t^S_{imk})D_{ik} + q_{imk} \quad \forall (i,m,k)\in P \tag{12}$$

$$h^E_{iMk} - (B_k - t^E_{iMk})D_{ik} \geq 0 \quad \forall i\in N,\ k\in K^T \tag{13}$$

$$h^S_{imk} = S_i - t^S_{imk}D_{ik} \quad \forall i\in N,\ m=1,\ k=1 \tag{14}$$

$$h^S_{imk} = h^E_{i,m-1,k} - (t^S_{imk} - t^E_{i,m-1,k})D_{ik} \quad \forall (i,m,k)\in P,\ m>1 \tag{15}$$

$$h^S_{imk} = h^E_{i,M,a(k)} - (B_{a(k)} - t^E_{i,M,a(k)})D_{i,a(k)} \\ - (t^S_{imk} - B_{a(k)})D_{ik} \ \ \forall i \in N, m = 1, k > 1 \tag{16}$$

$$\underline{S}_i \leq h^S_{imk},\ h^E_{imk} \leq \bar{S}_i \ \ \forall (i,m,k) \in P \tag{17}$$

In (1) we minimize the total expected cost. Constraints (5) and (6) result in a convex combination of schedule trees for each ship v, and to be valid must yield a single schedule tree. This can only happen if all schedule trees for ship v corresponding to $x_{sv} > 0$ follow the same tree of routes and the cost functions are linear over the convex hull (as the cost functions are in our examples). Constraint (2) calculates the number of occurrences of a port visit and ensures that each port visit occurs at most once. The variable y_{imk} is 0 if there are fewer than m ship visits at port i in scenario node k and is 1 otherwise. Constraint (3) calculates the loading or unloading quantity and constraint (4) calculates the start of service time for each port visit. If port visit (i,m,k) occurs, then the first term in (4) gives the start time for that service and the second term is zero. If port visit (i,m,k) does not occur, then the first term will be zero and the second term will be B_k, i.e., the end of the period for node k. Constraint (9) calculates the end of service time and (10) guarantees that there is no overlap between two services, i.e., a later port visit can only be served after the service of previous visit has been finished. Constraint (11) ensures that if a port is visited $m+1$ times in a scenario node, it must be visited m times in that scenario node. Constraints (12)–(17) are the inventory constraints. They ensure that the storage level is between the upper and lower bound of the storage tank at the start and end of each service. Since all flow rates are constant within a scenario node, the inventory level will change linearly between the start and end service times. So the constraints ensure that the inventory is within the bounds all the time within the whole planning period.

Reduced Cost

After a restricted master problem is solved (i.e., a master problem with a subset of the possible columns), dual variables will be known. These dual variables are denoted by d^A_{imk}, d^Q_{imk}, d^T_{imk}, and d^S_v for constraints (2)–(5), respectively. The reduced cost $\hat{C}_{sv}$ can then be calculated as follows:

$$\hat{C}_{sv} = C_{sv} - \sum_{i,m,k} (A_{svimk} d^A_{imk} + Q_{svimk} d^Q_{imk} + T_{svimk} d^T_{imk}) - d^S_v \\ = \sum_{(i,m,k)\to(i',m',k')\in E_s} P_k C_{ii'v} - \sum_{(i,m,k)\in N_s} (d^A_{imk} - d^Q_{imk} Q_{svimk} + d^T_{imk} T_{svimk}) - d^S_v \tag{18}$$

where P_k is the cumulative probability from start to node k that defines the scenario tree, E_s the set of edges, and N_s the set of port visits defining the tree of schedules

s, and $C_{ii'v}$ is the travelling cost along the edge $i \to i'$ for ship v. Constraint (18) expresses the reduced cost as the sum of terms over the edges and nodes in the tree of schedules.

Ship Routing Subproblems

The parameters Q_{svimk} and T_{svimk} as well as set E_s and N_s in (18) represent the route information generated by subproblems and are not given in advance. We wish to generate a column with the minimum reduced cost so we replace these parameters with variables q_{vimk} and t^S_{vimk} and also a variable route, which is specified by variable sets of edges E and nodes N that will define the schedule tree. For a ship v, the objective of the subproblem can be formulated as follows:

$$\bar{C}_v = \min_{E,N} \min_{q} \min_{t^S} \left(\sum_{(i,m,k)\to(i',m',k')\in E} P_k C_{ii'v} - \sum_{(i,m,k)\in N} (d^A_{imk} + d^Q_{imk} q_{vimk} + d^T_{imk} t^S_{vimk}) \right) - d^S_v \qquad (19)$$

In formulation (19), we try to find a physical visiting sequence and the corresponding values of q_{svimk} and t^S_{svimk} for each port visit in the sequence so as to minimize the reduced cost given in (18). The d^S_v term in (18) does not need to be considered in the subproblems. It can be subtracted from the objectives after solving the subproblems.

A ship subproblem can then be formulated as a shortest tree problem and solved by stochastic dynamic programming. The solution of the shortest tree problems is a tree of schedules with the least reduced cost, and yields a column that can be added into the master problem as a column. The state in the DP is (i,m,k,g,t), where i is the port, m is the order of the visit, k is the node of scenario tree, g is the amount of commodity on board the ship v when the ship arrives at port visit (i,m,k), and t is the start service time for the port visit (i,m,k). Both start service time, t, and the quantity on board the ship, g, are continuous quantities. However within the DP step they have to be restricted to discrete values, and this may lead to slight sub-optimality. If a service time is between two grid points, it will be delayed to the next grid point, and a regular grid is used for values of g so that there is no inaccuracy in accounting for the amounts on board ships. However, using discrete values for g and t does not mean that our model can only generate the solution with these discrete values. In fact, the master problem may choose several columns with the same physical tree of routes but different time and loading quantities and use the average of these columns as the solution, which may have the start service times and loading quantities different from discrete values.

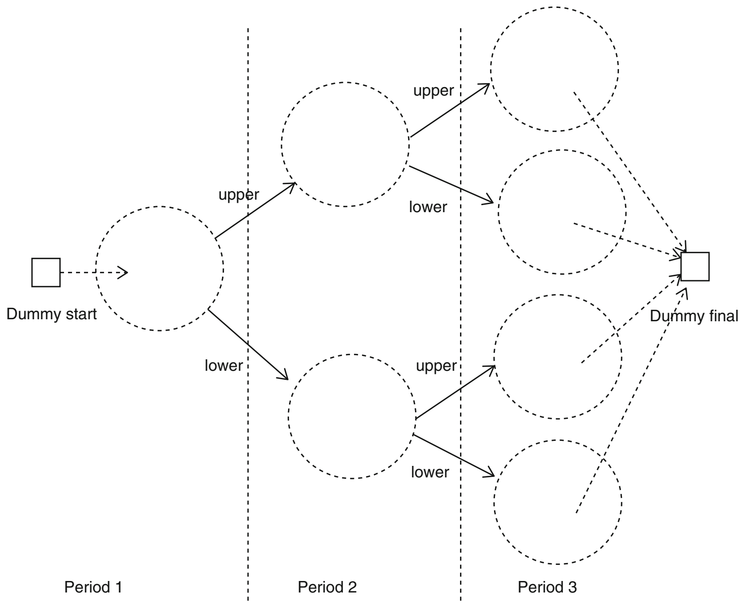

Fig. 1 Structure of a DP network showing demand scenario parts as *circles* and alternative upper and lower demand level branches in each period

Dynamic Programming Network

In this section, we describe the DP network for the ship subproblems. For each port visit in the network, there is a start service node and an end service node related to it. The costs in the objective are assigned to edges in the network. The DP network for a ship subproblem is related to the scenario tree which describes the pattern of consumer demands. We divide the network into several parts, each part, called a *demand scenario part,* represents a node in the demand scenario tree in the corresponding time period so that the DP network has the same top level structure as the demand scenario tree. See Figs. 1, 2, and 3 for examples.

In a DP network, a ship starts from the dummy start node, makes a set of port visits in different demand scenario parts of the network, and finishes the trip when it arrives at the dummy final node. When the ship is at a start service node, it makes decisions about how much to load or unload at the current port visit. When the ship is at an end service node it has a choice of three different actions: it can sail to another port visit in the same demand scenario part, it can stay at the current port visit until the future information is available before deciding which port to visit in the next period, or it can leave the current port visit immediately and sail to a port in the next period, in which case the future information about demand will be revealed during sailing but the ship will not change its destination port in response.

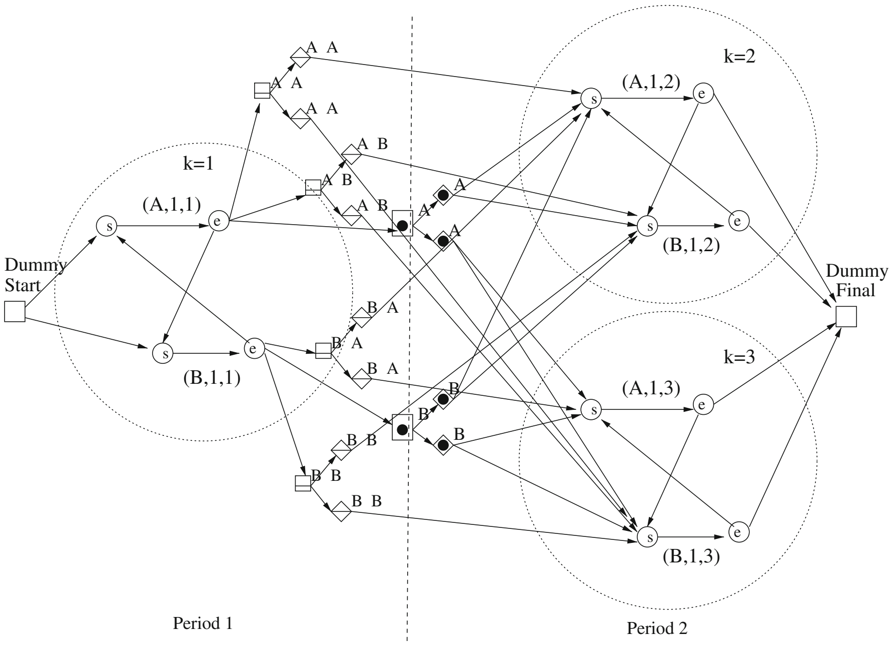

Fig. 2 DP network with two ports, A and B, and a maximum of one visit to each port in each demand scenario part

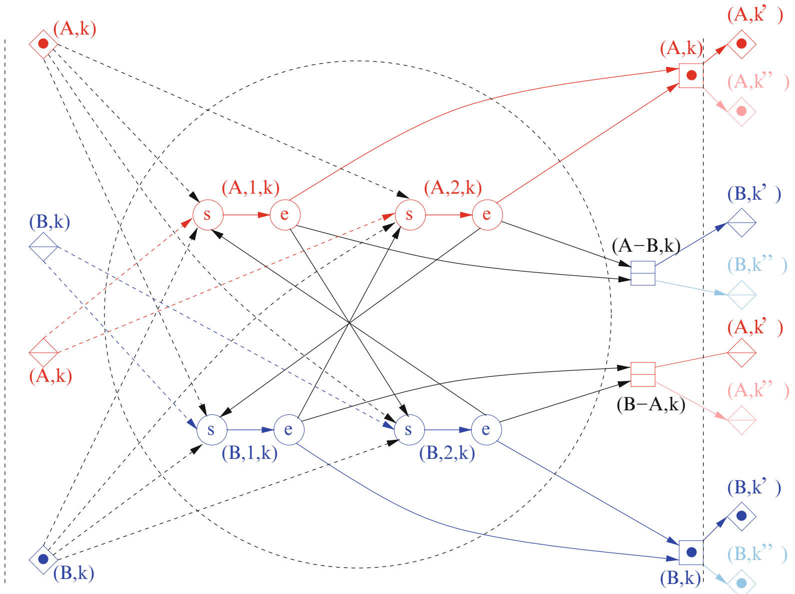

Fig. 3 Detailed demand part of DP network with two ports, A and B, and two possible port visits per port

Figure 2 is a simple example of a DP network with two time periods. There are three demand scenario parts in the network. The start node corresponds to the initial status of the ship. Its status is defined by its position (in some port or at a position at sea) and the amount of cargo on the ship. Figure 3 shows a fuller example of a single demand scenario part of a network.

Table 1 DP network node types

Node	Type	Description
Ⓢ	decision node	start service node: decision made is how much to load or unload during the port visit
ⓔ	decision node	end service node: decision made is the choice of next port visit or to delay until more information is available
⊟	sum-up node	forms the expected future value in port *i* at current time given the decision to sail to port *j* in next period
⊡	sum-up node	forms the expected future value in port *i* at end of current period before the decision of which port to visit next
◇	decision node	split node: decision at current time of which port visit of a given port to visit first in the next period
◈	decision node	split node: decision at end of current period of which port to visit in the next period and which is the first port visit for that port

The different types of nodes in DP networks are listed in the Table 1. Nodes Ⓢ, ⓔ and ◇ and the dummy start node are the decision nodes, and remaining nodes are sum-up nodes. Each port visit (i,m,k) has a start service node Ⓢ and an end service node ⓔ. For each boundary between two periods there is one ⊟$^{i-j}$ node for every pair of ports *i* and *j*. This is associated with a journey from port *i* to port *j* in a later period starting before the period boundary at which the demands in the next period will be revealed. Each ⊟$^{i-j}$ node is linked to a set of ◇$^{i-j}$ decision nodes, one for each demand scenario node in the following period, and its action is to sum up the expected costs at these nodes. Each ◇ is associated with one future demand scenario part and one port and selects the first port visit for that ship at that port. For example, in Fig. 2, a ship can go from end service node of port visit $(\mathrm{A},1,1)$ to sum-up node ⊟$^{\mathrm{A-B}}$ and sail to the start service node related to physical port B in period 2 through the split nodes. (However in this example there is only one port visit per port in period 2, so unlike the more general case shown in Fig. 3, there is no choice in this example at the split node.) The horizontal line inside the node is used to signify that the time window of this node is the whole period including the node.

Table 2 Edge costs and times in DP network

Edges	Edge costs	Min edge time
ⓢ $\to$ ⓔ	$-d^Q_{imk}(g-g') - d^T_{imk}t^S_{imk} - d^A_{imk}$	$\lvert g-g'\rvert/W_v$
◇ $\to$ ⓢ	$P_kC_{ii'v}$	$\tilde{T}_{ii'v}$
ⓔ $\to$ ⓢ	$P_kC_{ii'v}$	$\tilde{T}_{ii'v}$

Another sum-up node $\boxdot^{i}$ is on the boundary between two periods. This is associated with a journey from port i to any port in the next period after the demands in next period are known. Each $\boxdot^{i}$ node is linked to a set of ◈^{i} nodes, one for each demand scenario part. The decision made of which port to sail first at a ◈ node depends on the future demands in its scenario, so can be different in different demand scenario parts. The dot inside the node is used to signify that the arrival time at this node is fixed on the period boundary. In the DP network the decision nodes are ⓢ and ⓔ and ◇: ⓢ and ⓔ relate to decisions within the current demand scenario part and ◇ to the initial visit decisions in the following period.

Within a demand scenario part of the network, there can be edges from end service nodes ⓔ to start service nodes ⓢ. These edges are the travelling edges, and they have the associated travelling times and costs. Each end service node ⓔ (related to physical port i) is linked with several sum-up nodes $\boxminus^{i-j}$ and one sum-up node $\boxdot^{i}$. The port visits of the same physical port share the same sum-up nodes. For example, in Fig. 3, both end service nodes of port visit $(\mathrm{A},1,k)$ and $(\mathrm{A},2,k)$ are linked with sum-up nodes $\boxminus^{\mathrm{A-B}}$ and $\boxdot^{\mathrm{A}}$. Since a sum-up node $\boxdot$ is on the boundary between periods, its time is fixed so the edges from node ⓔ to node $\boxdot$ may have nonzero transition times on them. This corresponds to ships delaying their journeys until more demand information is available.

The edge costs in the DP network are derived from the objective function (19) of the ship routing subproblem, and are listed in the Table 2. Here g is the amount of commodity on board the ship when it arrives at start service node (i,m,k), while g' is the amount of commodity on board the ship at end service node (i,m,k). So the difference between them, $|g-g'|$, is the loading or unloading quantity in port visit (i,m,k). W_v is the (constant) loading or unloading rate for ship v, P_k is the cumulative probability of reaching node k in the demand scenario tree, $C_{ii'v}$ is the travelling cost from port i to port i' by ship v, and $\tilde{T}_{ii'v}$ is the (undelayed) sailing time from port i to port i'. Other edges which are in the network but not included in the above table have zero costs and zero minimum edge times and are used to define the stochastic structure of the network.

Every node in the network has a window for its visit time, so this is a stochastic DP problem with time windows. The time window for $\boxdot$ nodes is the single time of its period boundary, and the time window of every other node is initialized to be the same as the period in which it lies. However if there are more restrictive windows

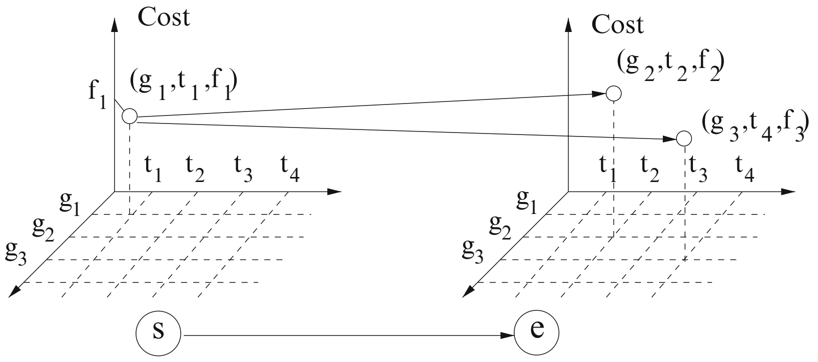

Fig. 4 Example of a service

for some nodes defined in the problem, the time windows on these nodes would be reduced, and all windows can also be reduced in the course of the solution by the B&B method.

Dynamic Programming Formulation

Since there are many different types of nodes in the DP network it is convenient to be able to refer to any node by a general single index, l. Each node l in the DP network for ship v has a function, denoted by $f_{lv}(t,g)$, giving the expected cost to the final node from node l when the node visit time is t and the amount on the ship is g. In our problem, there is a time window for the visit time at each node (i.e., $t \in [\tilde{A}_{lv}, \tilde{B}_{lv}]$). Since a sum-up node ⊡ is on the boundary between periods its time is fixed and so its time window has zero width and there is only one time point in its cost function, which therefore only depends on the quantity on board. Treating both t and g as continuous quantities makes the problem difficult to solve, so instead we restrict (t,g) to a discrete grid of values, $T \times G_v$. Consequently the loading quantity is also discrete as it is the difference in g between the start and end service nodes. An example of part of the $T \times G_v$ grid and a port service are shown in Fig. 4. This shows the expected cost f_1 in a start service node ⓢ at start of service time t_1 with quantity g_1 on board a ship, and the state reached in the end service node ⓔ for two of the possible loadings that will be considered in calculating the best value of f_1. For instance, point (t_4, g_3) corresponds to a service lasting $t_4 - t_1$ with the ship's load increased by $g_3 - g_1$.

The direction of solving stochastic dynamic programming is from dummy final node to the dummy start node. In the dummy final node L the cost function $f_{Lv}(t,g)$ is initialized to zero and on all other nodes is initialized to infinity (however, if we want to give a reward for a ship finishing early or having cargo on board at the end, then a more general $f_{Lv}(t,g)$ can be defined).

The values of $f_{lv}(t,g)$ are calculated recursively as follows, the calculations being for all ships v, all nodes l, all $t \in T \cap [\tilde{A}_{lv}, \tilde{B}_{lv}]$, and all $g \in G_v$.

- For l a Ⓢ start service node and l' the corresponding end service node ⓔ:

$$f_{lv}(t,g) = \min_{g' \in G_v : g' \geq g} \left\{ f_{l'v}(\tilde{T}_{lv}(t, g'-g),\ g') \ - d_l^Q(g'-g) - d_l^T t - d_l^A \right\},$$

  where $\tilde{T}_{lv}(t, \Delta g) = \min\{t' \in T : t' \geq t + |\Delta g|/W_v\}$ is the loading time rounded to the closest higher discrete time, and assuming node l corresponds to a visit to port i , $d_l^Q = d_i^Q$, $d_l^T - d_i^T$, and $d_l^A = d_i^A$ as given in Table 2. The recurrence above refers to a supply port. For a demand port the $g' \geq g$ is replaced by $g' \leq g$.
- For l a ◇ split node or ⓔ end service node:

$$f_{lv}(t,g) = \min_{l' : l \to l'} \ \min_{\max\{\tilde{A}_{l'v}, t + \tilde{T}_{ll'}\} \leq t' \leq \tilde{B}_{l'v}} \{f_{l'v}(t', g) + \tilde{C}_{ll'v}\},$$

  where if node l corresponds to a visit to port i, $\tilde{C}_{ll'v}$ is the cost of edge $l \to l'$ for ship v as shown in the Table 2, and $\tilde{T}_{ll'}$ is the transition time from a Ⓢ node l to a ⓔ node l', and for other cases is zero.
- For l a ⊟ or ⊡ sum-up node:

$$f_{lv}(t,g) = \sum_{l' : l \to l'} f_{l'v}(t,g)$$

The goal is to find the cost function $f_{Fv}(t)$ at the start dummy node F and the tree of schedules for the ship, which can be found by tracking forward from F.

Algorithm for Solving Subproblems

The most common algorithms for the shortest path problem with time windows are labelling algorithms, see [12, 13, 15]. These algorithms assign a label to each node in the network giving the cost of the currently known shortest path from the node to the final node. The algorithms repeatedly recalculate the labels for all the nodes in the network (in an order determined by some heuristic rules), until there is no improvement in the label of any node.

In this chapter we are considering a stochastic model whose objective is to minimize the expected cost. This requires the problem of finding a single shortest path which occurs in the deterministic case to be replaced by the problem of finding a tree of shortest paths. Each label associated with a node is now the lowest expected future cost known from the node to the end of the planning horizon for a specific node visit time and quantity on board. The set of all labels at a node therefore

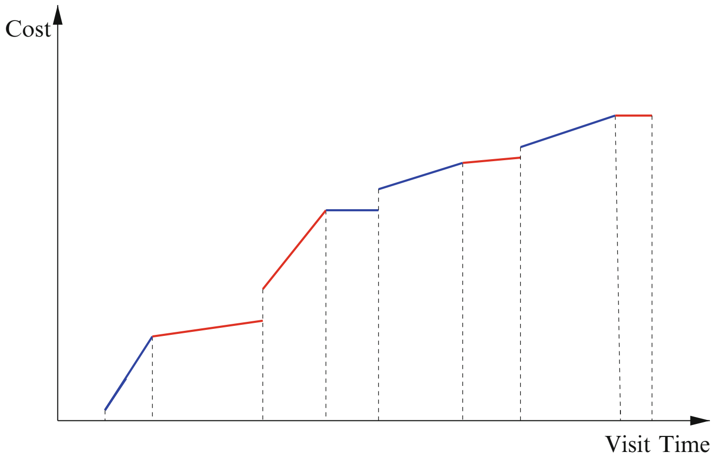

Fig. 5 Cost function

define this expected cost as a function of the visit time and quantity on board. An example of dependency of a cost function with visit time is shown in Fig. 5. The cost functions in our problem are increasing functions of time. In a deterministic DP network the shortest path calculation can be performed either from the start or from the terminal nodes. In the stochastic case where the expected costs are required the calculation starts at the terminal nodes. The iteration is started by setting the cost function to zero at this dummy final node and to infinity at all other nodes. The stochastic DP calculation then iteratively updates the cost function on each node in the network from all the nodes on its outgoing edges.

Because the graphs in our problems contain directed cycles, we may not be able to finish the updating by going through the network only once. We therefore have to update the node cost functions iteratively and be prepared to update the cost for one node several times. In an iteration of updating, we go through each node in the network, and for each node we consider all the outgoing edges from the node. If there has been any updating in the end node of an outgoing edge in last iteration, we will update the cost of the start node of the edge using the cost function of the end node. For the sum-up nodes, if one of the corresponding split nodes is updated in the previous iteration, the sum-up node will be updated in the current iteration. Therefore, we use a flag for each node to indicate whether or not the node is updated in the last DP iteration.

The number of iterations required during the updating is highly dependent on the order in which the nodes of the network are updated. Before starting to update the cost functions we order the nodes as follows. First we calculate the minimum number of directed edges from each node to the final dummy node. Then the nodes are ordered so that nodes closer to the final dummy vertex have lower index than those farther away. Then in each iteration the costs are updated in order of increasing node index. Once there has been no change in the cost of any node in a complete

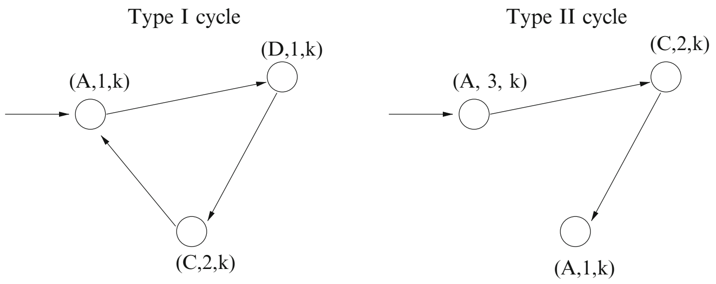

Fig. 6 Cycles of Type I and II

pass through all the nodes, then the optimal costs have been found and we choose the least cost from the cost function of the dummy start node in the network and track the shortest tree through the network from the dummy start node.

Cycle Elimination

Because there are several port visits for each port and the order of port visits to different ports is not predetermined by the structure of the network, it is possible to generate port visit paths which are not logically possible. There are two types of such impossible paths. Type I means that the ship returns to some port visit which occurred for that ship before, while Type II means that the ship route contains a port visit (i,m_1,k) and a later port visit (i,m_2,k) where $m_2 \leq m_1$. Examples of these two situations are shown in Fig. 6. We refer to both of these cases as cycles even though Type II may not include a cycle in the graph of port visits.

When there is a cycle of Type I, a port visit occurs more than once, and in this case the value of A_{svimk} will be greater than 1 and equal to the number of times the port visit occurs. Also the T_{svimk} and Q_{svimk} quantities will be the total over all the times the port visit occurs. Allowing cycles gives a relaxation of the true situation in our model, so bounds allowing them are still valid. However, a solution with cycles is not logically feasible. However allowing cycles gives a relaxation of the true situation, so we do not need to eliminate all the cycles when solving subproblems and can add the tree of routes including cycles into the master problem. Then we can eliminate cycles in the B&B algorithm by splitting a time window.

A K-cycle is defined to be a cycle of length K and elimination of K-cycles is well described in the literature, by Irnich and Villeneuve [22] and Irnich and Desaulniers [21]. However, it is not easy to avoid all the cycles with different lengths when solving the subproblems, and doing that is time consuming. We therefore only eliminate the 2-cycles. In our DP network, we divide a port visit into two nodes, a start service node and an end service node, so a 2-cycle in our problem is different from its original definition. See Fig. 7 for example. In the example, the start service and end service nodes for a port visit are regarded as a big node, and the definition of a 2-cycle is based on the corresponding port visits rather than the real nodes of the network.

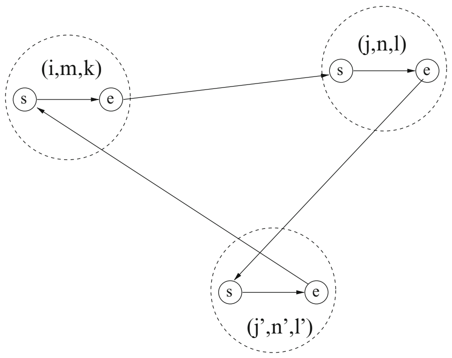

Fig. 7 A Type I cycle in our DP network

In our problem, cycles can only occur within a demand scenario part of the DP network, and there is no cycle crossing the period boundary or between different scenarios at the same period. Hence we only need to consider cycle elimination for start and end service nodes in the network. To eliminate 2-cycles using the method introduced in [22], at each Ⓢ and ⓔ node and for every time, we store the next port visit following the current one for the best and second best path for each time point. If there is a cycle when using the best solution, we use the second best solution instead. This usually allows us to avoid 2-cycles of Type I and II and we call paths satisfying this rule *allowed paths*. Note that all legal paths are allowed paths. We need to keep updating the best and second best solution on each node during the updating. The best path is the best among all allowed paths, while the second best is the best among all allowed paths where the next port visit is different from the best path.

Branch and Bound

The optimal solution of the stochastic ship routing problem must generate feasible schedules of all ships. However the master problem is a relaxation and may yield an infeasible ship schedule, either because the schedule is a mixture of schedules with different sequences of port visits or because it contains a cycle of port visits. To avoid this we use B&B. At each node of B&B tree a master problem with the discrete requirements relaxed is solved using column generation method. If the solution of this problem is not feasible, we branch so as to eliminate one of these infeasibilities. The columns generated from subproblems are kept in the master problem for other B&B nodes, only the infeasible column is deleted by setting the upper bound of the column to zero.

There are many possible choices for the branching strategy. We branch on infeasibilities in the following order.

If there are columns with positive weight in the solution that correspond to a path with a cycle, then we first branch on a time window so as to eliminate a cycle. Assume that port visit (i,m,k) is involved in a cycle. Let $\{t_{imk}^{S1},\dots,t_{imk}^{SK}\}$ be discrete start service times associated with the port visit (i,m,k). Let $\bar{t}_{imk} = 1/K \sum_{y=1\dots K} t_{imk}^{Sy}$ denote the average of these start service times. We do branching by splitting the time window $[\tilde{A},\tilde{B}]$ for the start service time of port visit (i,m,k). Since the width of the port visit time window is also reduced in child nodes, there is less chance of getting other cycles later in the solution.

If there are no cycles in the solution but there are fractional port visit variables, then a branch is made so as to either force a port visit to occur or not to occur. For a port i and node k, the set of port visit variables y_{imk} satisfies $y_{i1k} \geq y_{i2k} \geq y_{i3k} \geq \dots \geq y_{i,M-1,k} \geq y_{iMk}$ and to be feasible all values must be 0 or 1. We first calculate for each combination of (i,k) the difference between consecutive pairs of variables and choose the maximum difference:

$$Y_{i,k} = \max_{1\leq m\leq M-1} \{y_{i,m+1,k} - y_{i,m,k}\}$$

We then choose the minimum value for $Y_{i,k}$, and choose the maximum value of y_{imk} which is less than 1 and branch on that variable. If the chosen $y_{imk} \geq 0.5$, we branch first on $y_{imk} = 1$ and the other branch is $y_{imk} = 0$. If the value of chosen $y_{imk} < 0.5$, we branch first on $y_{imk} = 0$ and the other branch is $y_{imk} = 1$.

When in a branch where $y_{im'k}$ is set to 0, no port arrivals (i,m,k) can occur for $m \geq m'$. So we delete all the port arrivals (i,m,k) (where $m \geq m'$) as well as all the edges linked with these port arrivals from the network of each ship. If $y_{im'k}$ is set to 1 in a branch, no update happens for the structure of the ship networks. However, a small artificial negative cost is added to each edge from the start service node of port visit (i,m',k) to its end service node, which makes port visit (i,m',k) more attractive than port arrivals (i,m,k) for $m \geq m'$.

If there are no cycles or non-integer y_{imk}, then we calculate the flow $x_{imkjnlv}$, where $x_{imkjnlv} = \sum_{s\in R_v;(i,m,k)\to(j,n,l)\in E_s} x_{sv}$. This quantity defines whether or not ship v sails from port visit (i,m,k) to port visit (j,n,l). For each (j,n,l), we find the maximum fractional value for $x_{imkjnlv}$. Then from these maximum values we choose the minimum value over (j,n,l). The formulation for this process is shown as follows:

$$\min_{j,n,l} \max_{i,m,k,v} \{x_{imkjnlv}\}$$

If the value of the chosen variable is less than 0.5, we branch first on $x_{imkjnlv} = 0$ and $x_{imkjnlv} = 1$ on the other branch. In the branch where $x_{imkjnlv}$ is set equal to 0, the ship v does not sail from (i,m,k) to (j,n,l). Hence all corresponding edges are deleted from the network of ship v. In the branch where $x_{imkjnlv}$ set to 1, we delete all the edges for ship v coming out of (i,m,k) except those going into (j,n,l). For all other ships, the edges from (i,m,k) to (j,n,l) are deleted from the networks.

Table 3 Properties of test examples

EX	Ports	Max arrival	Scenario nodes in tree	Planning periods	Branches	Ships
a1	3	2	3	2	2	2
b1	5	3	3	2	2	2
b2	5	3	3	2	2	2
b3	5	3	3	2	2	2
c1	5	3	7	3	2	2
c2	5	3	7	3	2	2
c3	5	3	7	3	2	2
d1	6	4	7	3	2	3
d2	6	4	7	3	2	3
d3	6	4	7	3	2	3
f1	5	3	13	3	3	2
f2	5	3	13	3	3	2
g1	6	3	13	3	3	3
g2	6	3	13	3	3	3
g3	6	3	13	3	3	3
h1	8	4	40	4	3	3
h2	8	4	40	4	3	3

Stochastic ship routing problems are computationally demanding, and as a result there is the danger that the B&B search may terminate because of time or memory limits before finding an acceptable feasible solution. Depth-first search, although not the fastest B&B search strategy for proving optimality, has the advantage of finding feasible solutions early. Best-first B&B algorithm is a better strategy for proving optimality, and both strategies can be combined by first using depth-first search to find an early integer solution and then switching to best-first search to produce better bounds. This mixed strategy worked well on some examples; however, all the results reported in the next session use depth-first search only and were solved to zero gap.

Examples and Results

To test the models and solution methods developed in this chapter, a set of test problems has been built. The implementation is written in C and CPLEX10.0 is used to solve the sequence of LPs in each B&B node of the master problem. The ship subproblems are independent of each other and are solved in parallel using OpenMP on a 4-core processor. The data structures needed to represent the networks in the subproblems are generated once only before the start of the optimization.

Table 3 gives the characteristics of each test problem. Example $a1$ is very small and is used to illustrate the details of a solution, including the visit sequences, start service time, quantity on board each ship, and the storage levels. All of these

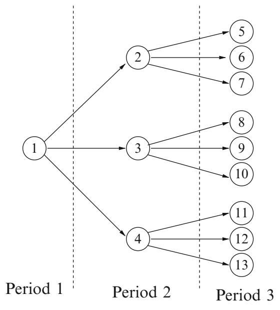

Fig. 8 Scenario tree of Example g1

Table 4 DP and master problem dimensions

EX	Nodes	Edges	(i,m,k) Combinations	Constraints
a1	56	82	18	152
b1	137	706	45	372
b2	137	706	45	372
b3	137	706	45	372
c1	347	1786	105	862
c2	347	1786	105	862
c3	347	1786	105	862
d1	416	2335	126	1033
d2	416	2335	126	1033
d3	416	2335	126	1033
f1	632	3421	195	1607
f2	632	3421	195	1607
g1	758	3421	234	1928
g2	758	4477	234	1928
g3	758	4477	234	1928
h1	3170	23,481	960	7898
h2	3170	23,481	960	7898

details are given as an example later in this section. The examples named with the same first letter are problems with the same physical ports layout and the same demand scenario tree structure, but different initial inventory levels and demand rate situations at each port. The "Max Arrival" column gives the maximum number of possible arrivals for each port in each demand scenario part, which is the parameter M in the formulation introduced in section "Master Problem". The "Scenario Nodes", "Planning Periods" and "Branches" columns give the structure of the scenario tree. For example, in example g1, there are 13 demand scenario parts, 3 time periods, and 3 branches for each period in the scenario tree, which indicates a scenario tree as shown in Fig. 8 (Table 4).

Table 5 Computational results (elapsed time in sec)

EX	B&B nodes	Columns	Total time	Subprob time	Master iters
a1	6	56	0.8	0.6	24
b1	78	1251	13	11	497
b2	177	3079	31	25	1407
b3	219	4204	47	41	1973
c1	81	2435	20	18	879
c2	87	3948	26	21	1633
c3	237	4757	57	48	1978
d1	564	6206	120	103	2649
d2	63	1353	15	14	284
d3	750	6945	138	105	2954
f1	405	9034	439	379	3799
f2	138	3623	126	118	1181
g1	342	7241	403	352	2805
g2	624	11,557	705	611	4731
g3	132	4109	181	161	1298
h1	3598	30,753	3690	3112	43,850
h2	2987	31,983	3371	2958	40,791

The computational results given in Table 5 are: the number of branch-and-bound nodes used to find the optimal discrete solution, the total number of columns generated from the subproblems, the total solving time, the elapsed time for solving the subproblems, and the total number of column generation iterations in the master problem. Examples a1–c3 are relatively small and can be solved within a minute. However, when the problem size is increased, the solving times for the later examples increase sharply. Another factor which may effect the solving time is the initial storage levels and demand situations. For instance, examples f1 and f2 have the same problem structure, but different initial storage levels and demand situations, and f2 is solved much faster than f1. This is because the initial storage levels and demand situations are related to the number of visits to each port in each demand scenario part. If there is sufficient initial storage at a port, fewer visits may be required, which reduces the length of the visiting sequences for ships and makes the problem easier to solve.

As previously discussed, because of the size of the DP networks, the major solving time in each example is taken in solving the ship subproblems, and Table 5 indicates that this takes around 75–94 % of the total time. In the tests the ship subproblems are solved in parallel using one core per ship.

Some detailed solutions are given based on two of the above examples. In example c1, there are 5 ports, and ports A, B, and C are customer ports and ports D and E are supply ports. The left-hand side of Fig. 9 shows the demand scenario tree of the example, and the demand trend changes in each demand scenario part. The tree of routes on the right-hand side of Fig. 9 shows the ship routes in the solution of c1. In the figure, ships choose different routes according to the different demand situations in each period. For instance, ship 1 visits the different ports in the upper and lower cases of period 2, since in the upper case the demands at ports A and B go up while

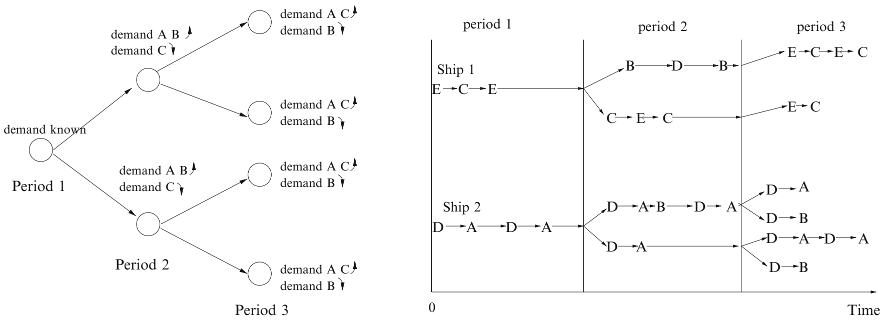

Fig. 9 Solution of Example c1

the demand at port C goes down, and in the lower case the demand situations are just the opposite. In period 3, ship 1 does nothing in the lower case, and this is because all of the demands are satisfied in the case so that there is no need to travel any further.

Figure 10 shows the optimal solution for b1. The physical routes, inventory levels, and quantities on board ships are shown. The changes in the storage of each consumer port and on ships as a function of time can be clearly seen. In period 1, ship 1 sails the route D→A→D. There is an unloading service made by the ship at port A so that there is an increase in the storage level at port A. There are also two visits made by ship 2 to port C, so the storage level of port C goes up twice during the period. There is no visit to port B for the whole period, and the stock level of port B goes down throughout the period because of the constant demand rate. A similar situation can be seen in period 2 from the same figure.

Conclusion

In this chapter, we propose a solution approach to solve the stochastic ship routing problem with inventory management at the ports. The only uncertainty considered is the demand levels at the ports. A Branch and Price algorithm is presented. A master problem is formulated as a set partitioning model including inventory constraints, while a subproblem for each ship is solved by dynamic programming to find the least reduced cost columns for the master problem. The optimal integer solution is searched along the B&B tree and column generation method is used to solve the relaxed LP iteratively in each B&B node.

The ship routing subproblems are stochastic dynamic programming problems, and they are solved by a backward labelling algorithm. The method we use is analogous to the methods that have been used in the deterministic case, but have had to be extended to deal with the scenario branching in the stochastic case. The minimum expected costs from the start node to the final dummy node is calculated.

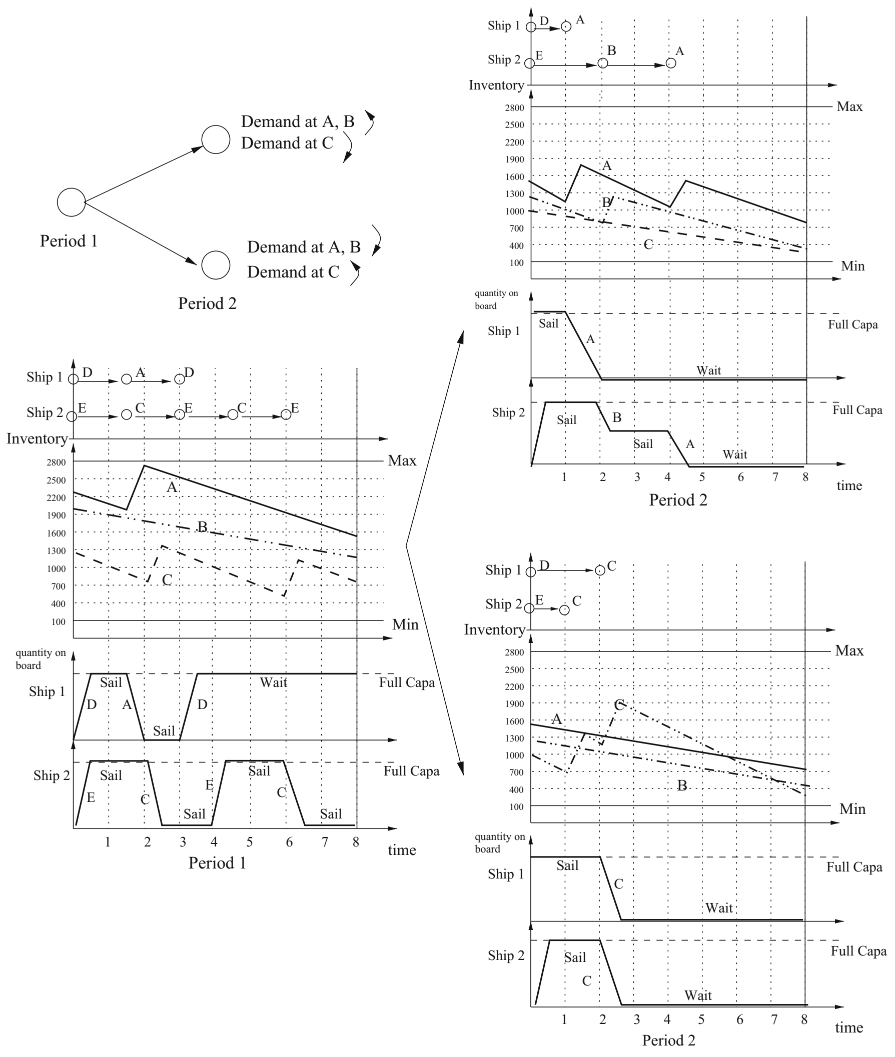

Fig. 10 Solution of Example b1

Because of the complicated DP network, there are many possible cycles (which are not feasible in a solution). Two-cycles are eliminated when solving the subproblems and other cycles with length greater than 2 are eliminated during the B&B algorithm by splitting the time windows. Because the ship subproblems are independent of each other, OpenMP is used to solve them in parallel on a multi-core computer.

From the computational experience, our decomposition method is able to solve medium sized examples. A set of test examples with different geographical port layouts, number of ships, scenario trees, and initial storage situations were built and were solved by the decomposition method. Our computational experience shows that around 75–94 % of the elapsed time to solve the problem is used to solve the ship subproblems, even when subproblems are solved in parallel. The rest of the elapsed time is used to do B&B administration and solve the LPs. Because of the need to model on entire scenario tree, the stochastic problems become large, even for a small transport network, and this limits the size of problem that can be solved. Generating useful columns in a heuristic way a priori is a possible area for further work. The generated columns can be added into the master problem to give a warm start, which should reduce the solution times and allow larger problems to be solved.

The methods in this chapter naturally extend to cases where ships can divert during sailing (when new demand information becomes available) and cases where ships can alter their speed. These cases give rise to nonlinear subproblems (with whole problem becoming a stochastic nonlinear integer programming problem). However because the subproblems can be solved by discretization and DP the solution approach given in this chapter can still be applied.

References

1. Appelgren, L.: A column generation algorithm for a ship scheduling problem. Transp. Sci. **3**, 53–68 (1969)
2. Appelgren, L.: Integer programming methods for a vessel scheduling problem. Transp. Sci. **5**, 64–78 (1971)
3. Bendall, H., Stent, A.: A scheduling model for a high speed containership service: a hub and spoke short-sea application. J. Marit. Econ. **3**(3), 262–277 (2001)
4. Bertsimas, D.: A vehicle routing problem with stochastic demand. Oper. Res. **40**(3), 574–585 (1992)
5. Christiansen, M.: Decomposition of a combined inventory and time constrained ship routing problem. Transp. Sci. **33**(1), 3–16 (1999)
6. Christiansen, M., Fagerholt, K.: Robust ship scheduling with multiple time windows. Nav. Res. Logist. **49**, 611–625 (2002)
7. Christiansen, C., Lysgaard, J.: A branch-and-bound algorithm for the capacitated vehicle routing problem with stochastic demands. Oper. Res. Lett. **35**, 773–781 (2007)
8. Christiansen, M., Nygreen, B.: A method for solving ship routing problems with inventory constraints. Ann. Oper. Res. **81**, 357–378 (1998)
9. Christiansen, M., Nygreen, B.: Modelling path flows for a combined ship routing and inventory management problem. Ann. Oper. Res. **82**, 391–412 (1998)
10. Christiansen, M., Fagerholt, K., Ronen, D.: Ship routing and scheduling: status and perspectives. Transp. Sci. **38**(1), 1–18 (2004)
11. Crary, M., Nozick, L., Whitaker, L.: Sizing the U.S. destroyer fleet. Eur. J. Oper. Res. **136**, 680–695 (2002)
12. Desrochers, M., Soumis, F.: A generalized permanent labeling algorithm for the shortest path problem with time windows. INFOR **26**(3), 191–211 (1988)
13. Desrochers, M., Soumis, F.: A reoptimization algorithm for the shortest path problem with time windows. Eur. J. Oper. Res. **35**, 242–254 (1988)
14. Desrochers, M., Desrosiers, J., Solomon, M.: A new optimization algorithm for the vehicle routing problem with time windows. Oper. Res. **40**, 342–354 (1992)

15. Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F.: Time constrained routing and scheduling. In: Network Routing. Handbooks in Operations Research and Management Science, vol. 8, pp. 35–139. North-Holland, Amsterdam (1995)
16. Dror, M., Trudeau, P.: Stochastic vehicle routing with modified saving algorithm. Eur. J. Oper. Res. **23**, 228–235 (1986)
17. Dror, M., Laporte, G., Trudeau, P.: Vehicle routing with stochastic demands: properties and solution frameworks. Transp. Sci. **23**(3), 166–176 (1989)
18. Gendreau, M., Laporte, G., Seguin, R.: An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transp. Sci. **29**(2), 143–156 (1995)
19. Gunnarsson, H., Ronnqvist, M., Carlsson, D.: A combined terminal location and ship routing problem. J. Oper. Res. Soc. **57**, 928–938 (2006)
20. Hjorring, C., Holt, J.: New optimality cuts for a single-vehicle stochastic routing problem. Ann. Oper. Res. **86**, 569–584 (1999)
21. Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. Les Cahiers du GERAD G-2004-11 (2004)
22. Irnich, S., Villeneuve, D.: The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. INFORMS J. Comput. **18**(3), 391–406 (2006)
23. Kleywegt, A., Nori, V., Savelsbergh, M.: Dynamic programming approximations for a stochastic inventory routing problem. Transp. Sci. **38**, 42–70 (2004)
24. Mehrez, A., Hung, M., Ahn, B.: An industrial ocean-cargo shipping problem. Decis. Sci. **26**(3), 395–423 (1995)
25. Ronen, D.: Marine inventory routing: shipments planning. J. Oper. Res. Soc. **53**, 108–114 (2002)
26. Sherali, H., Al-Yahoob, S., Hassan, M.: Fleet management models and algorithms for an oil-tanker routing and scheduling problem. IIE Trans. **31**, 395–406 (1999)
27. Shih, L.H.: Planning of fuel coal imports using a mixed integer programming method. Int. J. Prod. Econ. **51**, 243–249 (1997)