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

Ken McKinnon Yu Yu
School of Mathematics, University of Edinburgh

8th April 2011

Abstract

This paper 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 in the deterministic problem, and in the stochastic problem, the demand rate for a period is not known until the beginning of that period. The demand situation in each time period can be 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 yeild 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.

1 Introduction

The marine shipping industry has experienced an unprecedented boom over the past decade. Not only because of the rapid growth of the requirements of the 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.

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 Christiansen et al. 2004. This focuses on literature about ship routing and scheduling published between year of 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 Ronen 2002 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. 1999 have presented a MIP model for the Kuwait Petroleum Corporation (KPC) problem. Because of the integrality conditions and 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. 2002 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 Bendall and Stent 2001, Mehrez et al. 1995 and Shih 1997. Heuristics are developed in Gunnarsson et al. 2006 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. 1992 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 Appelgren 1969 and Appelgren 1971 describe a typical tramp ship scheduling problem, which was the first work 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 Christiansen and Nygreen 1998a,

Christiansen and Nygreen 1998b and Christiansen 1999, 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 paper is closest to that used in their paper. Compared to their papers, the present paper deals with different inventory situation, 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. Christiansen and Lysgaard 2007 use a branch-and-price algorithm for vehicle routing problem with stochastic demands. In their 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. Dror and Trudeau 1986 use a straight-forward modification of the Clark and Wright savings algorithm for the SVRP based on a discussion of route failure. In Hjorring and Holt 1999 and Gendreau et al. 1995, an integer L-shaped method is used to solve SVRP to optimality. Bertsimas 1992 proposes a priori sequence among all customers of minimal expected total length, 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 Dror et al. 1989.

There are few references in the literature to stochastic ship routing problems with inventory. Christiansen and Fagerholt 2002 presents robust ship scheduling with multiple time windows. A set partitioning approach with the columns found a priori is proposed 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 Kleywegt et al. 2004, and approximation methods are used to find acceptable solutions.

This paper 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 constists 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 based method that uses Dantzig-Wolfe (DW) decompostion 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 labeling method.

The structure of the rest of the paper is as follows. Section 2 introduces the Dantzig-Wolfe decomposition approach and describes the structure of the master and sub-problems. This section also describes the techniques used to eliminate cycles. Section 3 gives the Branch and Bound algorithm. Section 4 presents computational results and Section 5 gives the conclusion.

2 Decomposition Approach for the Stochastic Ship Routing Problem

2.1 Assumptions

The ocean transportation problem is too complex to consider every factor in the real world when modeling the problem. To simplify the problem, the following assumptions are made before introducing the detailed model.

- 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. This, however, is not a limitation as the service can continue without a break in the following period.

2.2 Solution Framework

A branch-and-price algorithm is used in this paper. 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 are an infinite number of these columns, but the DW approach generates only a finite number of them. In each iteration of DW a subproblem is solved for each ship to generate an attractive tree of schedules

for that ship. In this paper 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 ungenerated columns. 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 smallest 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.

2.3 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 index 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 – m-th visit at port i in node k
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} – 1 if ship v makes port visit (i,m,k) in tree of schedules s, 0 otherwise
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) (-ve corresponds to loading) if ship makes that port visit in schedule tree s, and 0 otherwise
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 the node k
W_i – unloading rate from ship to port i (-ve corresponds to 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 (-ve at a supply port)
$\underline{S_i}$ – initial stock level in port i
$\overline{S_i}$ – 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 they are zeros or non-zeros at the same time. If port visit (i,m,k) is made by ship v in schedule tree s, parameter A_{svimk} is 1. And parameters Q_{svimk} and T_{svimk} represent the quantity loaded and the start service time for this port visit respectively. But A_{svimk} could be an integer value greater than 1 if there are cycles involved in the solution.

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) (-ve corresponds to 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

$$\min \sum_{v\in V}\sum_{s\in R_v} C_{sv}x_{sv} \tag{2.1}$$

$$\sum_{v\in V}\sum_{s\in R_v} A_{svimk}x_{sv} = y_{imk} \ \ \forall (i,m,k)\in P \tag{2.2}$$

$$\sum_{v\in V}\sum_{s\in R_v} Q_{svimk}x_{sv} = q_{imk} \ \ \forall (i,m,k)\in P \tag{2.3}$$

$$\sum_{v\in V}\sum_{s\in R_v} T_{svimk}x_{sv} + (1-y_{imk})B_k = t^S_{imk} \ \ \forall (i,m,k)\in P \tag{2.4}$$

$$\sum_{s\in R_v} x_{sv} = 1 \ \ \forall v\in V \tag{2.5}$$

$$x_{sv}\geq 0 \ \ \forall v\in V, s\in R_v \tag{2.6}$$

$$\{x_{sv} : s\in R_v\} \text{ yield a valid tree of schedules for ship } v, \ \forall v \tag{2.7}$$

$$y_{imk}\in\{0,1\} \ \ \forall\ (i,m,k)\in P \tag{2.8}$$

$$t^E_{imk} = t^S_{imk} + q_{imk}/W_i \ \ \forall (i,m,k)\in P \tag{2.9}$$

$$t^E_{i,m-1,k} \leq t^S_{imk} \ \ \forall (i,m,k)\in P,\ m>1 \tag{2.10}$$

$$y_{imk}\geq y_{i,m+1,k} \ \ \forall (i,m,k)\in P \tag{2.11}$$

$$h^E_{imk} = h^S_{imk} - (t^E_{imk} - t^S_{imk})D_{ik} + q_{imk} \ \ \forall (i,m,k)\in P \tag{2.12}$$

$$h^E_{iMk} - (B_k - t^E_{iMk})D_{ik} \geq 0 \ \ \forall i\in N,\ k\in K^T \tag{2.13}$$

$$h^S_{imk} = S_i - t^S_{imk}D_{ik} \ \ \forall i\in N,\ m=1,\ k=1 \tag{2.14}$$

$$h^S_{imk} = h^E_{i,m-1,k} - (t^S_{imk} - t^E_{i,m-1,k})D_{ik} \ \ \forall (i,m,k)\in P\ m>1 \tag{2.15}$$

$$\begin{aligned} h^S_{imk} = {} & h^E_{i,M,a(k)} - (B_{a(k)} - t^E_{i,M,a(k)})D_{i,a(k)} \\ & - (t^S_{imk} - B_{a(k)})D_{ik} \ \ \forall i\in N, m=1, k>1 \end{aligned} \tag{2.16}$$

$$\underline{S_i} \leq h^S_{imk},\ h^E_{imk} \leq \overline{S_i} \ \ \forall (i,m,k)\in P \tag{2.17}$$

In (2.1) we minimize the total expected costs. Constraints (2.5) and (2.6) result in a convex combination of schedule trees for each ship v. To be valid this convex combination must be the same as 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. Constraints (2.2) calculates 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 (2.3) calculates the loading or unloading quantity and constraint (2.4) calculates the start of service time for each port visit. If port visit (i,m,k) occurs, then the first term in (2.4) 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 (2.9) calculates the end of service time and

constraint (2.10) 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 (2.11) 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 (2.12)-(2.17) 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.

2.4 Reduced Cost

After a restricted master problem is solved, 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 (2.2)–(2.5) respectively. The reduced cost $\hat{C}_{sv}$ can then be calculated as following:

$$\hat{C}_{sv} = C_{sv} - \sum_{i,m,k} (A_{svimk} d^A_{imk} + Q_{svimk} d^Q_{imk} + T_{svimk} d^T_{imk}) - d^S_v \tag{2.18}$$

$$= \sum_{(i,m,k)-(i',m',k') \in E_s} P_k C_{ii'v} - \sum_{(i,m,k) \in N_s} (d^A_{imk} - d^Q_{imk} Q_{svimk} + d^T_{imk} T_{svimk}) - d^S_v \tag{2.19}$$

where P_k is the cumulative probability from start to node k in the scenario tree, E_s the a set of edges included in tree of schedules s, N_s is a set of port visits included in tree of schedules s and $C_{ii'v}$ is the traveling cost along the edge $i \rightarrow i'$ for ship v.

Equation (refoo) expresses the reduced cost as the sum of terms over the edges and nodes in the tree of scedules.

2.5 Ship Routing Subproblems

The parameters Q_{svimk} and T_{svimk} as well as set E_s and N_s in (2.19) represent the route information generated by subproblems and is 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 the sets edges E and nodes N in the schedule tree. For a ship v, the objective of the subproblem can be formulated as following:

$$\bar{C_v} = \min_{E,N} \min_{q} \min_{t^S} [\sum_{(i,m,k)\to(i',m',k')\in E} P_k C_{ii'v} - \sum_{(i,m,k)\in N} (d^A_{imk} + d^Q_{imk} q_{vimk} + d^T_{imk} t^S_{vimk})] - d^S_v \qquad (2.20)$$

In formulation 2.20, 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 presented in formula 2.19. d^S_v in 2.19 does not need to be considered in the subproblems. It can be subtracted from the objectives after solving the subproblems.

A ship subproblem then can 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 port, m is the m-th visit, k is the node of scenario tree, g is the amount of commodity on board the ship v when the ship arrives the port visit (i,m,k), and t is the start service time for the port visit (i,m,k). Both start service time and quantity on board the ship are continuous quantities. In practice, we use discrete quantities for both g and t so as to allow a discrete version of DP to be used. A regular grid is used for the discrete start service time t. If a start service time is between two grid points, it will be delayed to the next grid point. 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.

2.5.1 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. And we allocate the costs in the objective into different 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 demand scenario part, represents a scenario node in the corresponding time period so that the DP network has the same structure as the scenario tree. See Figure 1 as an example.

In a DP network, a ship starts from the dummy start node, visits a set of port visits in different demand scenario parts of the network, and finishes the

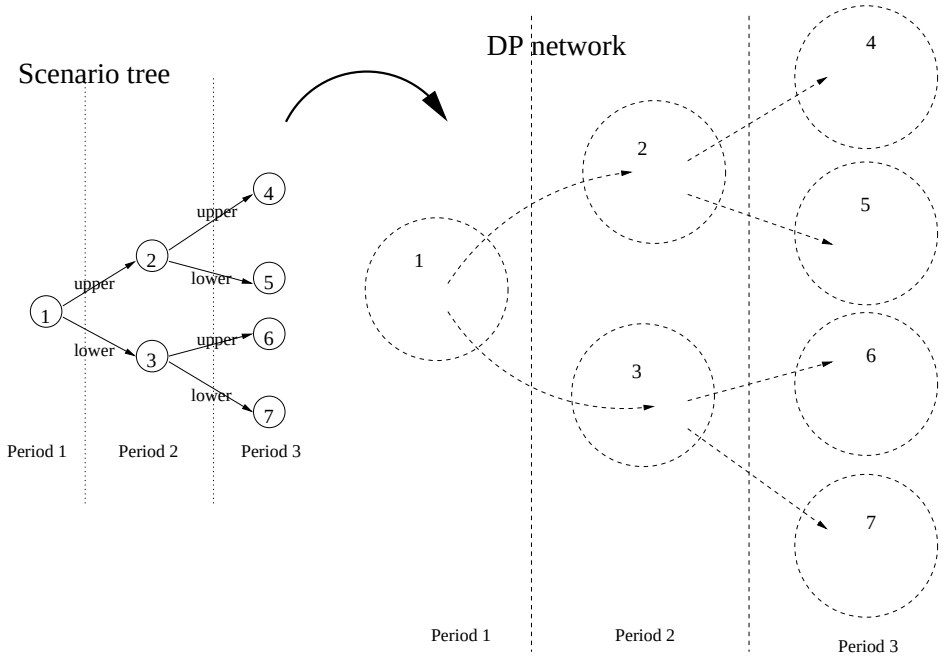

Figure 1: Structure of DP network

trip when it arrives at the dummy final node. Within a demand scenario part, if the ship is on a start service node, it makes decisions about how much to load or unload at the current port visit. And when the ship is on an end service node, it has choices of three different actions: it can sail to another port visit in the same demand scenario part and do another service to the port visit, it can leave the current port visit immediately and sail to the port visits in the demand scenario parts of the next period, or it can delay at the current port visit until the future information is available. We will introduce the nodes and arcs in the DP network which are associated with these ship actions later in this section.

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. The different types of nodes in the network are listed in the table below:

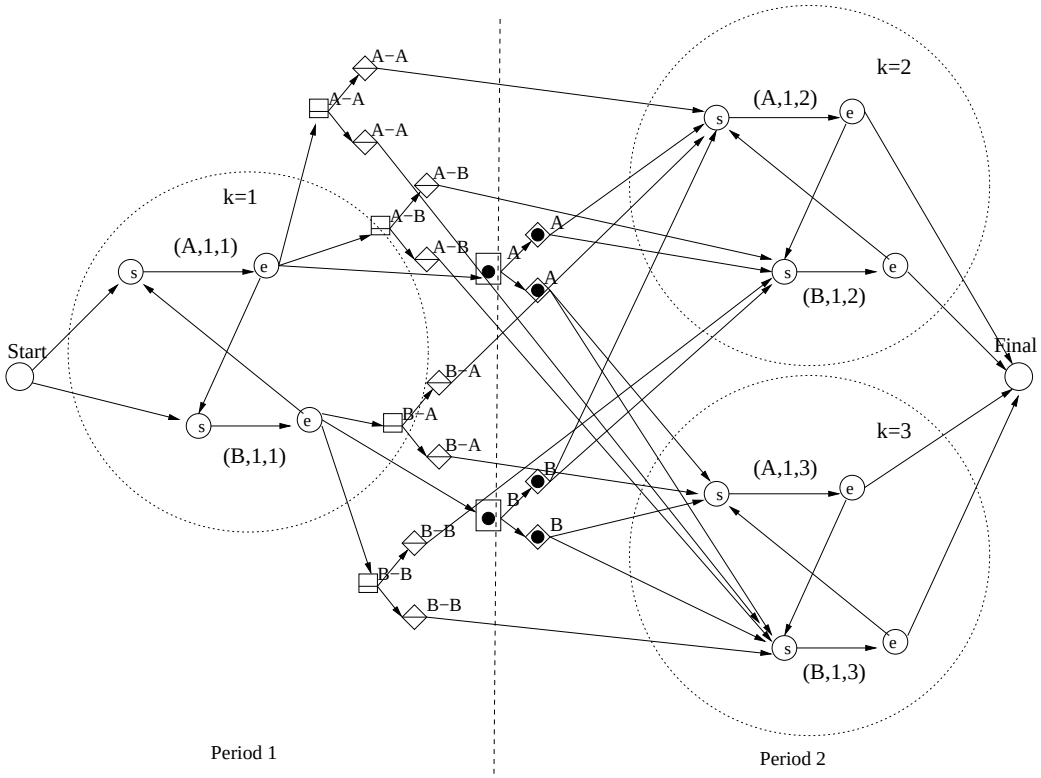

Figure 2: DP network with maximum of one visit to each port in each demand scenario part

Node	Type	Description
ⓢ	decision node	start service node: decision made on the node is to choose how much to load or unload during the port visit
ⓔ	decision node	end service node: decision made is to choose next port visit or decision to delay until more information is available
$\boxminus^{i-j}$	sum-up node	sum-up node: expected value of sailing at current time from port i in current period to port j in next period
$\boxdot^{i}$	sum-up node	sum-up node: expected value of delaying sailing from port i to the end of the current period
◇	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

As shown in the table, node ⓢ and ⓔ are both decision nodes in the DP network. 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 $\boxminus^{i-j}$

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 where the next period demand information is revealed. Each $\boxminus^{i-j}$ node is linked to a set of $\diamondsuit$ decision nodes, one for each demand scenario node in the following period. Each $\diamondsuit$ is associated with one future demand scenario part and one port and selects the first port visit for that port. For example, a ship can go from end service node of port visit $(A,1,1)$ to sum-up node $\boxminus^{A-B}$ and sail to the start service node related to physical port B in period 2 through the split nodes. The horizontal line inside the node means that the time window of this node is the whole period including the node. Another sum-up node on the boundary between two periods is $\boxdot^{i}$. This is associated with a journey from port i to any port in a later period after the demand information for next period is known. Again each $\boxdot^{i}$ node is linked to a set of $\diamondsuit$ nodes, one for each demand scenario part. The decisions made on these $\diamondsuit$ nodes can be different from each other according to the known demand situations. The dot inside the node indicates that the arrival time to this node is fixed on the period boundary. In the DP network both $\diamond$ and $\bigcirc$ nodes are decision nodes. We use different symbols here to tell whether a decision is made to visit a node within the current period or a node in future period. The details about how these nodes relate to different ship actions in the problem are given in the later.

Within demand scenario part of the DP network, a ship arrives at a start service node ⓢ, starts loading or unloading, finishes the service at node ⓔ and sails to other nodes. In Figure 2, each node relates different discrete values for the quantities on board ship when it arrives at the node and a grid of start service time points so that the cost function on each node has three dimensions. An example of the cost function is given in Figure 3.

Since there are a group of discrete quantities and time points on each node, a service is decided by the time points and quantities on both start and end service node. See Figure 4 for example. In the example, we have a point in the cost function of a start service node ⓢ and it relates to the time t_1 and quantity g_1 on board a ship. Then three different points in the cost function of an end service node ⓔ give three different service situations. For instance, point (t_4, g_3) means a service lasting $t_4 - t_1$ with a loading or unloading quantity $|g_3 - g_1|$.

Within a scenario part of the network, there are arcs from end service node ⓔ to start service node ⓢ. These arcs are the traveling arcs, and there are traveling times and traveling costs related on these arcs. 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 the network both end service nodes of port visit $(A,1,k)$ and $(A,2,k)$ are linked with sum-up node $\boxminus^{A-B}$, $\boxminus^{A-A}$ and $\boxdot^{A}$. There is no transition time on the arcs from node ⓔ to node $\boxminus^{i-j}$ so that

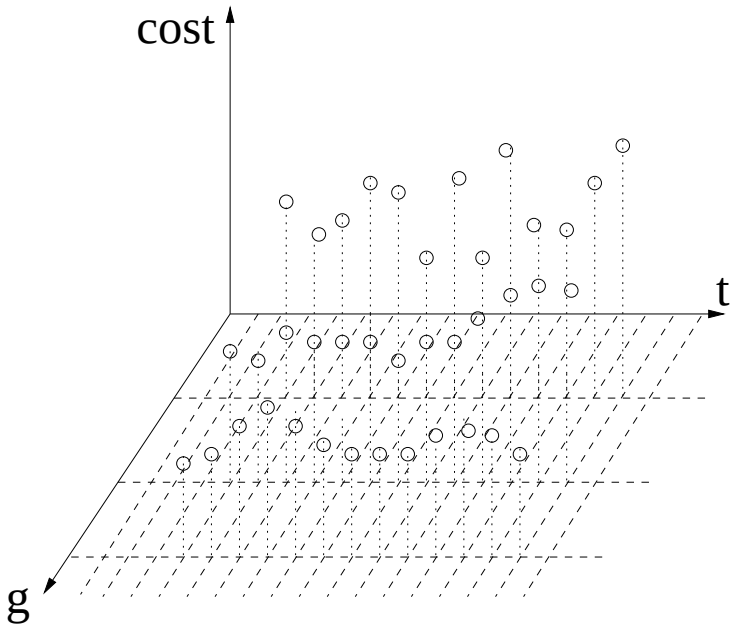

Figure 3: General cost function on node

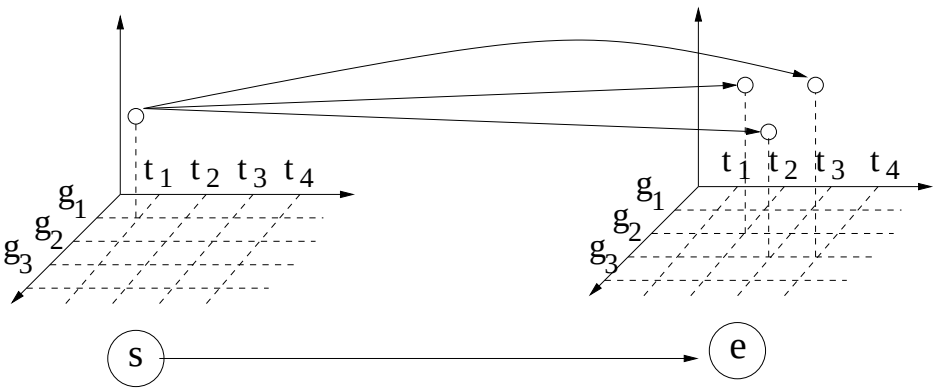

Figure 4: Example of a service

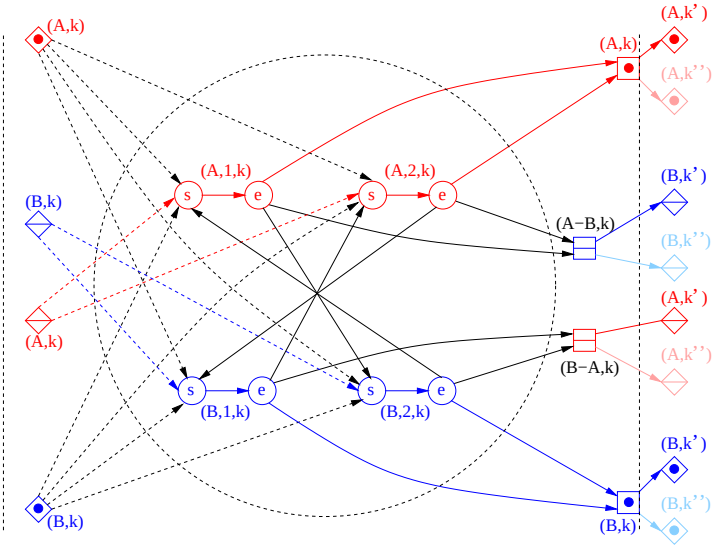

Figure 5: Detailed DP network with different end service nodes within a part

the time grids on both nodes are the same. While sum-up node $\boxdot^{i}$ is on the boundary of a period and there is only one time point in the cost function. Hence the arcs from node ⓔ to node $\boxdot^{i}$ may have nonzero transition times on them. There are also a set of arcs linking the end service node ⓔ with the final dummy node in the network. These arcs are needed when solving the DP along the network.

The DP network shown in Figure 2 looks simple, but because each node contains all the possible combination of discrete t and g, we may have to consider a large number of situations of t and g when updating the cost function along an arc in the network. An alternative way to build the DP network is to use different nodes for different discrete quantities on board ship (g) instead of using a single node associated with a group of discrete quantities on the node. Figure 5 shows an example DP network by using different nodes for different quantities g within a demand scenario part of DP network. A start service node links to several different end service nodes with different quantities, and each link between the start service node to one of the end service nodes relates to a loading or unloading service. In this type of DP network, the end service nodes related to the same physical port and the same quantity on board the ship will share the same sum-up nodes. For instance, we can see from Figure 5 that end service node of $(B,1,1,g_1)$ and $(B,2,1,g_1)$ are linked with the same sum-up nodes $\boxminus^{B-B}$, $\boxminus^{B-A}$ and $\boxdot^{B}$. This DP network is used when we solve the subproblems in practice, and later we will build DP recurrence formulation based on this kind of network.

According to the objective function of the ship routing subproblem, the edge costs in the DP network are given in the following table:

Edges	Edge Costs	Edge Time
ⓢ → ⓔ	$-d^Q_{imk}\lvert g^E - g^S\rvert - d^T_{imk}t^S_{imk} - d^A_{imk}$	$\lvert g^E - g^S\rvert / W$
$\diamond \rightarrow$ ⓢ	$P_k C_{ii'v}$	travel time
ⓔ → ⓢ	$P_k C_{ii'v}$	travel time

In the table, g^S is the amount of commodity on board the ship when it arrives at start service node (i,m,k), while g^E is the amount of commodity on board the ship at end service node (i,m,k). So the difference between them $|g^E - g^S|$ is the loading or unloading quantity in port visit (i,m,k). W is the loading or unloading rate for the ship which is a constant. P_k is the cumulative probability of reaching node k in the scenario tree. $C_{ii'v}$ is the traveling cost from port i to port i' by ship v. Other edges which are in the network but not included in the above table have zero costs and are used to build the stochastic structure of the network.

This is a stochastic DP problem with time windows, because each node in the network has a time window for start of service. Initially, the time window of a node is the full time period but it can be reduced by the Branch and Bound method when solving the problem.

2.5.2 Dynamic Programming Formulation

The direction of solving stochastic dynamic programming is from dummy final node to the dummy start node. The solution can then be tracked from the start dummy node. In the networks of our ship subproblems, there are several different types of nodes: start service nodes ⓢ, end service nodes ⓔ, sum-up nodes $\boxminus^{i-j}$ and $\boxdot^{i}$, and split nodes $\diamond$. These nodes are indexed in different ways, so in order to avoid having to write different DP recurrence relation for each possible transition we shall introduce a single index for each node in the network. If this is denoted by l and then the recursive formula $f_{lv}(t)$ is the least expected cost from node l at time t to the final node in the network. In our problem, there is a time window for the start service time for each node (i.e. $t \in [\tilde{A}_{lv}, \tilde{B}_{lv}]$). The value on the final dummy node $f_{Lv}(t)$ is set to zero. However, if we want to give a reward for a ship finishing early, then an increasing function can be used for $f_{Lv}(t)$. The detail DP formulation is given below.

For decision nodes (ⓢ, ⓔ and $\diamond$) the recurrence formulas for ship v are:

- For start service node ⓢ:

$$f_{lv}(t) \quad = \quad \min_{l':l\rightarrow l'} \{ f_{l'v}(t + \tilde{T}_{ll'}) + \tilde{C}_{ll'v} \}, \quad t \in [\tilde{A}_{lv}, \tilde{B}_{lv}] \quad (2.21)$$

- For split node $\diamond$:

$$f_{lv}(t) = \min_{l':l\rightarrow l'} \min_{\max\{\tilde{A}_{l'v}, t+\tilde{T}_{ll'}\} \le \tau \le \tilde{B}_{l'v}} \{ f_{l'v}(\tau) + \tilde{C}_{ll'v} \}, \; t \in [\tilde{A}_{lv}, \tilde{B}_{lv}] (2.22)$$

- For end service node ⓔ:

$$f_{lv}(t) = \min\{\min_{l':l\to l'} \min_{\max\{\tilde{A}_{l'v},t+\tilde{T}_{ll'}\}\le\tau\le\tilde{B}_{l'v}} \{f_{l'v}(\tau)+\tilde{C}_{ll'v}\}, \min_{l':l\to l'}\{f_{l'v}(t+\tilde{T}_{ll'})+\tilde{C}_{ll'v}\}\}, \quad t\in[\tilde{A}_{lv},\tilde{B}_{lv}] \tag{2.23}$$

$\tilde{A}_{lv}$ and $\tilde{B}_{lv}$ are the lower and upper bounds of the time window at node l. $\tilde{C}_{ll'v}$ is the cost of edge $l\to l'$ for ship v, which is shown in the table in the previous section. $\tilde{T}_{ll'}$ is the transition time from a ⓢ node l to a ⓔ node l', and for other cases is the minimum time for the transition. In formula 2.23, if l' is a ⓢ node we use the first item, while if l' is a ⓔ node, we use the second item.

When l is a sum-up node $\boxminus^{i-j}$ or $\boxdot^{i}$, the cost function is:

$$f_{lv}(t)=\sum_{l':l\to l'} f_{l'v}(t), \quad t\in[\tilde{A}_{lv},\tilde{B}_{lv}] \tag{2.24}$$

We want to find the cost function at start dummy node $f_{l_0v}(t)$, where l_0 is the dummy start node, according to above DP recurrence formulations.

2.5.3 Algorithm for solving subproblems

In literature, the algorithms for the shortest path problem with time windows usually assign pairs of labels to each time in each node. Each node in the network is associated a label, which consists of a label for the cost of the path to the actual node and a label for the visit time at the node. The algorithms update these labels for each node through the network according to the dominance rules iteratively until there is no improvement can be made for any node. This is called a Labeling Algorithm in literature. Literature Desrochers and Soumis 1988a, Desrochers and Soumis 1988b and Desrosiers et al. 1995 give the details about these algorithms.

As a stochastic model is built here, we want to find the shortest tree through the network from the dummy start node to the final node. Each time within a node has a label which is the lowest expected cost known from that node to the final dummy node. By deleting the final node and the edges into it we can get a tree. This is the shortest "tree" problem with time windows. The shortest tree problem with time windows, can be solved by stochastic dynamic programming, and generalizes the shortest path problem with time windows. Cost on each node is the minimum expected cost from the current node to the final node as a function of the time of reaching the current node. An example of a cost function is shown in Figure 6. The cost functions in our problem are increasing functions. In our DP network, there is a known start node and a known final node, so in the deterministic case we have the

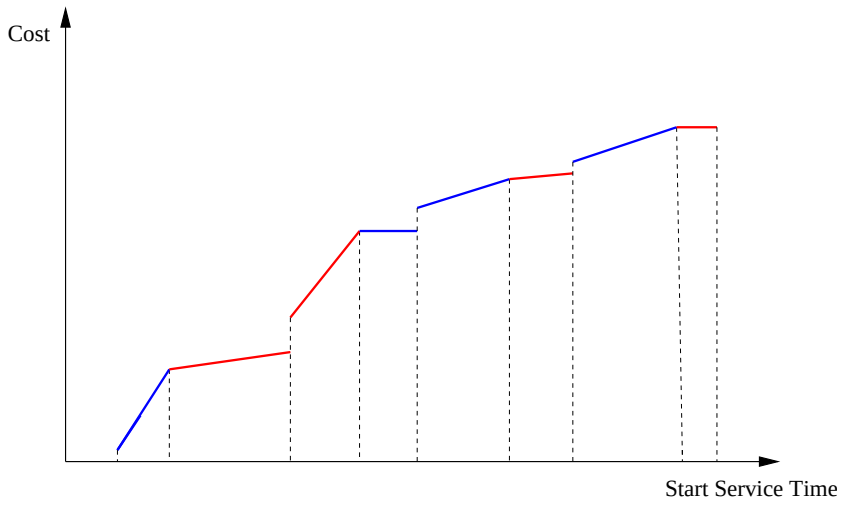

Figure 6: Cost function

choice of calculating costs from the initial node or costs from the final node. In the stochastic case however we need to calculate the expected costs to the final node. We therefore start the iteration by setting the value of the final node to 0 and all other nodes to infinity, and update the cost function on each node in the network from the nodes on its outgoing arcs.

When solving the DP problem, we need to update each node in the network using the DP formulation given in last section. Because the graph contains directed cycles, we may not be able to finish the updating by going through the network only once. So we have to update the node costs iteratively and prepare 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 arcs from the node. If there has been any updating in the end node of an outgoing arc in last iteration, we will update the cost of the start node of the arc 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 (for any time) in the last DP iteration.

We do the updating through the DP network for each node in order from smaller index to bigger index. The number of iterations required during the updating is highly depended on the order in which the nodes of the network are updated. Before updating cost functions through network, we need to first find an order for nodes in the network and we do this as follows. In our network, the minimum number of steps from each node to the final dummy node is first calculated and an order of nodes is decided by checking the numbers: the node with a smaller minimum number of steps to the final will

get a smaller node index, which means it will be considered in an early order during the updating.

Once we have updated cost function for all the nodes and there have been no changes, then the optimal costs have been found and we choose the least cost from the cost function of the start dummy node in the network and track the shortest tree through the network.

3 Branch and Bound

The optimal solution of the stochastic ship routing problem must satisfy the discrete restrictions relating to the relation of single routes for each ship. Branch and Bound algorithm is used here to search for feasible discrete solutions. At each node of Branch and Bound tree a problem with the discrete requirements relaxed is solved using column generation method. If the solution of the problem does not satisfy the discrete constraints or includes a cycle, we branch so as to eliminate one of these infeasibilities. The columns generated from subproblems are kept in the master problem for other Branch and Bound nodes, only the infeasible column is deleted by setting the upper bound of the column to zero. There are a lot of ways to decide branching strategies. We do branching on the fractional variables 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 $\{t_{imk}^{S1},...,t_{imk}^{SK}\}$ be discrete start service times associated with the port visit (i,m,k). Let $\bar{t}_{imk} = 1/K \sum_{y=1..K} t_{imk}^{Sy}$ denote the average of these start service times. We do branching by splitting the time window $[a,b]$ 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 $y_{i1k} \geq y_{i2k} \geq y_{i3k} \geq ... \geq y_{i,M-1,k} \geq y_{iMk}$ 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 arrival (i,m,k) (where $m \geq m'$) as well as all the edges linked with these port arrival 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, an artificial negative cost is added to each edge from start service node of port visit (i,m',k) to end service node of (i,m',k), which makes port visit (i,m',k) more attractive and more likely be included in the solution of the corresponding subproblem.

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 the 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 $x_{imkjnlv} = 1$ in 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 arcs for ship v coming out of (i,m,k) except those going into (j,n,l). For all other ships, the arcs from (i,m,k) to (j,n,l) are deleted from the networks.

Depth first branch-and-bound algorithm is not the fastest strategy if we wish to prove optimality. However, because of the problem size of the stochastic ship routing problem, we may fail to find the integer solution before reach the memory or solving time limits. Hence when solving the problem, we use Depth First branch-and-bound algorithm here so as to find a feasible integer solution earlier. However, Best First branch-and-bound algorithm is known as the fastest searching strategy. Therefore, we can combine these two branch-and-bound strategies together, using the first Depth First Search to find an early integer solution and then switch to the Best First Search so that we can finish the searching earlier. This mixed strategy was tried on some examples and was effective, however the results below are for the depth first search case.

4 Examples and Results

To test the models and solution methods developed in this paper, 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 Branch and Bound node of the master problem. The ship subproblems are independent of each other and are solved in parallel using OpenMP. The structure of networks of subproblems are generated a priori and input as data.

Table 1 gives the characteristics of each test problem.

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

Table 1: Example Information

In table 1, $a1$ is a very small problem. This example was built to demonstrate the details of the solution, including the visit sequences, start service time, quantity on board each ship, the storage levels, and so on. All of these 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 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 scenario tree node, which is the parameter M in the formulation introduced before. 'Scenario Nodes', 'Planning Periods' and 'Branches' columns give the structure of the scenario tree. For example, in example $g1$, there are 13 scenario tree nodes, 3 time periods and 3 branches each period in the scenario tree, which indicates a scenario tree as shown in Figure 7.

In the stochastic ship routing problem, we use the combinations of (port, arrival, scenario node) as the state of the problem. For each port visit

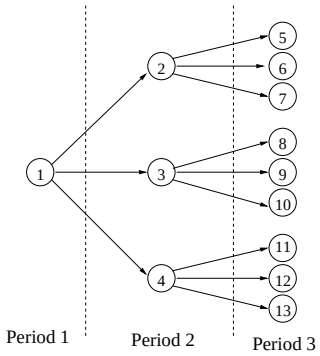

Figure 7: Scenario tree of Ex. g1

(i,m,k), the start service time of the port visit should be within a time window. In all the examples here, we set the initial time window for each port visit (i,m,k) to be the full time period of the scenario tree node k. In the dynamic programming problem for ship subproblems, there are several nodes related to a port visit, for instance start service node, end service node, two types of sum-up nodes and splitting nodes, as discussed in Section 2.5.1. There are also a set of links between two nodes in the DP network. We allow ships to travel between supply port visit and consumer port visit, and travels between two close consumer ports within the same scenario tree node. There are arcs linking the sum-up nodes and the splitting nodes as well as linking the splitting nodes to the start service nodes. This information is given in Table 2. The table also shows the number of (i,m,k) combinations and the number of constraints in the master problem.

The computational results are shown in Table 3.

Table 3 gives 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 to 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, example $f1$ 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 scenario tree node. 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

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	23481	960	7898
h2	3170	23481	960	7898

Table 2: DP and Master Problem Dimensions

major solving time in each example is used to solve the ship subproblems, and Table 3 indicates that around 75% – 93% of the total time is used solving the subproblems. Here we solve subproblems in a parallel way so as to reduce the total solving for the subproblems.

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 Figure 8 shows the scenario tree of the example, and the demand trend changes in each scenario tree node. The tree of routes on the right hand side of Figure 8 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 demand for port A and B goes up while the demand for 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 9 shows the optimal solution for example 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

EX	B&B nodes	Columns	time total	time sub	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	11557	705	611	4731
g3	132	4109	181	161	1298
h1	3598	30753	3690	3112	43850
h2	2987	31983	3371	2958	40791

Table 3: Computational Results without Tolerance

be clearly seen. In period 1, ship 1 sails the route $D \rightarrow A \rightarrow D$. There is a 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.

5 Conclusion

In this paper, we propose a solution approach to solve stochastic ship routing problem with inventory management problem. In the problem, demand is the only uncertainty. A Branch and Price algorithm is presented in the paper. 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 Branch and Bound tree and column generation method is used to solve the relaxed LP iteratively in each Branch and Bound node.

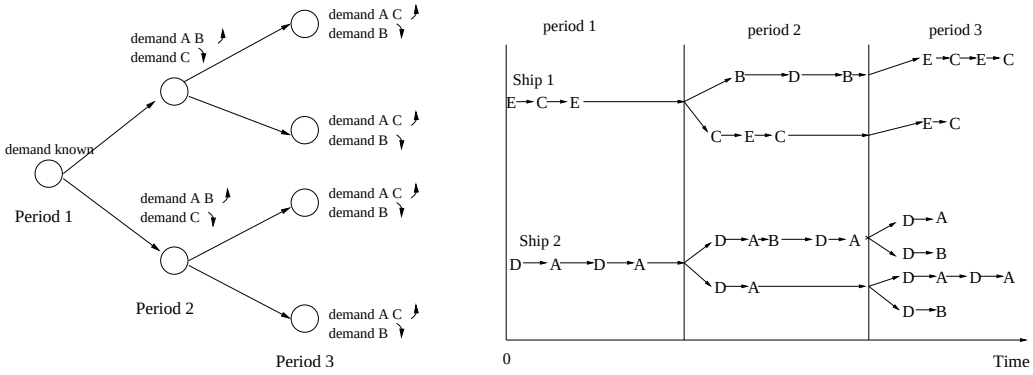

Figure 8: Solution c1

To solve the ship routing subproblems, a backward set labeling algorithm is used to solve the stochastic dynamic programming problem. 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. Because of the complicated DP network, there are many possible cycles (which are not feasible in a solution). 2-cycles are eliminated when solving the subproblems and other cycles with length greater than 2 are eliminated during the Branch and Bound algorithm by splitting the time windows. Because the ship subproblems are independent of each other, OpenMP is used to solve these subproblems 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 tree and initial storage situations were built and were solved by the decomposition method. Our computational experience shows that around 75% – 93% of the elapsed time to solve the problem is used to solve the ship subproblems, even when examples are solved in parallel. The rest of the elapsed time was used to do Branch and Bound administration and solve the LPs. We cannot however solve large problems. Because of the need to model on entire scenario tree, the stochastic problems become large, even for a small transport network.

For the future work, an alternative model, which allows diverting cases during sailing for ships, can be explored based on the model given in the paper. Generating useful columns in a heuristic way a priori is another possible further work. The generated columns can be added into the master problem as initial columns so that we can solve the problem with a warm start, which can help us solve the problem quickly.

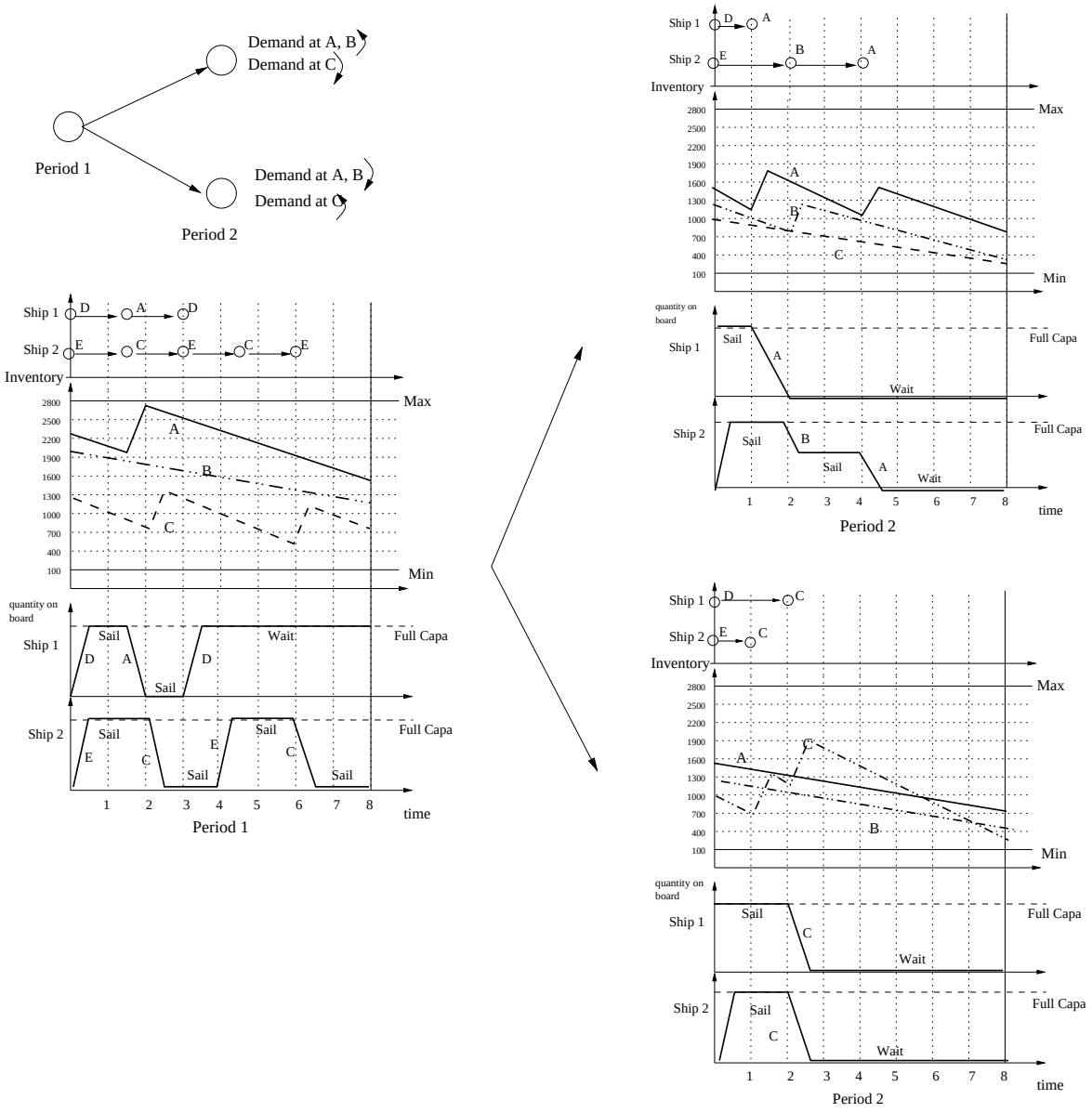

Figure 9: Solution Example b1

References

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

Appelgren, L. (1971). Integer programming methods for a vessel scheduling problem. *Transportation Science*, 5:64–78.

Bendall, H. and Stent, A. (2001). A scheduling model for a high speed containership service: A hub and spoke short-sea application. *Journal Maritime Economics*, 3(3):262–277.

Bertsimas, D. (1992). A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585.

Christiansen, C. and Lysgaard, J. (2007). A branch-and-bound algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35:773–781.

Christiansen, M. (1999). Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33(1):3–16.

Christiansen, M. and Fagerholt, K. (2002). Robust ship scheduling with multiple time windows. *Naval Research Logistics*, 49:611–625.

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

Christiansen, M. and Nygreen, B. (1998a). A method for solving ship routing problems with inventory constraints. *Annals of Operations Research*, 81:357–378.

Christiansen, M. and Nygreen, B. (1998b). Modelling path flows for a combined ship routing and inventory management problem. *Annals of Operations Research*, 82:391–412.

Crary, M., Nozick, L., and Whitaker, L. (2002). Sizing the u.s. destroyer fleet. *European Journal of Operational Research*, 136:680–695.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354.

Desrochers, M. and Soumis, F. (1988a). A generalized permanent labeling algorithm for the shortest path problem with time windows. *INFOR*, 26(3):191–211.

Desrochers, M. and Soumis, F. (1988b). A reoptimization algorithm for the shortest path problem with time windows. *European Journal of Operational Research*, 35:242–254.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. *Handbooks in Operations Research and Management Science 8, Network Routing, North-Holland, Amsterdam*, pages 35–139.

Dror, M., Laporte, G., and Trudeau, P. (1989). Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science*, 23(3).

Dror, M. and Trudeau, P. (1986). Stochastic vehicle routing with modified saving algorithm. *European Journal of Operational Research*, 23:228–235.

Gendreau, M., Laporte, G., and Seguin, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29(2):143–156.

Gunnarsson, H., Ronnqvist, M., and Carlsson, D. (2006). A combined terminal location and ship routing problem. *Journal of the Operational Research Society*, 57:928–938.

Hjorring, C. and Holt, J. (1999). New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research*, 86:569–584.

Kleywegt, A., Nori, V., and Savelsbergh, M. (2004). Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Science*, 38:42–70.

Mehrez, A., Hung, M., and Ahn, B. (1995). An industial ocean-cargo shipping problem. *Decision Science*, 26(3):395–423.

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

Sherali, H., Al-Yahoob, S., and Hassan, M. (1999). Fleet management models and algorithms for an oil-tanker routing and scheduling problem. *IIE Transactions*, 31:395–406.

Shih, L.-H. (1997). Planning of fuel coal imports using a mixed integer programming method. *Internat. Journal of Production Economics*, 51:243–249.