Stochastic Ship Fleet Routing with Inventory Limits

YU YU

Doctor of Philosophy
University of Edinburgh
2009

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*YU YU*)

First of all, I would like to express my deepest gratitude to my supervisor, Professor Ken Mckinnon, for his patient guidance throughout my Ph.D study, his enthusiasm, and immense knowledge. He is not only a supervisor of the Ph.D study, but also an advisor of my life. This thesis would never have been finished without his support.

I would like to thank Maija and Ken for letting me feel Edinburgh as my second home town. I can never forget the happiest time spent with you at the Christmas and new year eve. I would like to thank Danny for reading the draft and giving many valuable comments. I am grateful to all my friends here for bringing me lots of wonderful moments during the many years I stayed in Edinburgh.

Finally, my special thanks go to my parents and my wife, Changxu, for their constant understanding, encouragement and support as family.

Edinburgh, November 2009

Yu Yu

Abstract

This thesis describes a stochastic ship routing problem with inventory management. The problem involves finding a set of least costs routes for a fleet of ships transporting a single commodity when the demand for the commodity is uncertain. Storage at consumption and supply 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.

Several possible solution approaches for solving the problem are studied in the thesis. This problem can be formulated as a mixed integer programming (MIP) model. However solving the problem this way is very time consuming even for a deterministic problem with small problem size. In order to solve the stochastic problem, we develop a decomposition formulation and solve it using a Branch and Price framework. A master problem (set partitioning with extra inventory constraints) is built, and the subproblems, one for each ship, involve solving stochastic dynamic programming problems to generate columns for the master problem. Each column corresponds to one possible tree of schedules for one ship giving the schedule for the ship for all demand scenarios. In each branch-and-bound node, the node problem is solved by iterating between the master problem and the subproblems. Dual variables can be obtained solving the master problem and are used in the subproblems to generate the most promising columns for the master problem. Computational results are given showing that medium sized problems can be solved successfully.

Several extensions to the original model are developed, including a variable speed model, a diverting model, and a model which allows ships to do extra tasks in return for a bonus. Possible solution approaches for solving the variable speed and the diverting model are presented and computational results are given.

Contents

Chapter 1

Introduction

The marine shipping industry has experienced an unprecedented boom over the past decade. Not only because of the rapid growth of the international economy, which requires the transfer of more and more energy and commercial commodities from one country 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.

Transportation problems have been extensively discussed in literature. The classical routing and scheduling problem for vehicles and ships is an 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. 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. From heuristic perspective, metaheuristics have been proven to be a good approach for quickly solving large real world problems and obtaining high quality, near optimal, solutions.

The ship routing problem (SRP) and vehicle routing problem (VRP) are two particular cases of the routing problems dealing with transportation. Both of them are hard combinatorial problems and not easy to solve. In the vehicle routing problem, a fleet of vehicles, starting and ending at a single depot, are considered to cover a set of customers within a period so that the traveling cost is minimized and all the customer demands are satisfied. In VRP, we usually consider one type of commodity and inventory is not included in the problem. However, in the ship routing problem, a fleet of ships are traveling among ports to delivery commodities. There is no particular depot for each ship. Inventory at each port is an important factor in the problem. Among solution approaches for the ship routing problem, the set partitioning model with columns generated heuristically and a priori plays an important role in the literature. In this solution approach, different heuristics are developed

with the aim of producing good quality ship schedules which can be used in the set partitioning models. Although mixed integer programming (MIP) is another possible tool for the problem, because of the integrality requirement of the model, it is time-consuming to solve the MIP model directly. For this reason heuristic methods or dynamic programming algorithms are always involved in the MIP solution approach in order to find a good solution more efficiently. Dantig-Wolfe decomposition approach has been widely used for the routing problem, since this provides a similar lower bound of the problem as that found by the Lagrangian Relaxation, and this guarantees an efficient search in branch-and-bound algorithm. Christiansen and Nygreen 1998a, Christiansen and Nygreen 1998b, Christiansen 1999 and Christiansen 1996 have presented the Dantzig-Wolfe decomposition approach for deterministic ship routing problem with inventory constraints, and a comprehensive survey of solution approaches for the ship routing problem is provided by Christiansen et al. 2004.

In the Dantzig-Wolfe decomposition approach, subproblems are usually formulated as the shortest path problem with time windows (SPPTW), and solving the subproblems quickly is very important for the solution approach. Ioachim et al. 1994 present a dynamic programming algorithm for the SPPTW with linear node costs, and Desrochers and Soumis 1988a give a generalized permanent labeling algorithm for the SPPTW. The same authors also propose a re-optimization algorithm for the same problem in Desrochers and Soumis 1988b. Add these methods are proven to be efficient ways to solve SPPTW.

More recently, ship routing problems with uncertainties have been investigated. Since there are a big number of unpredictable factors involved in the ocean shipping industry, a deterministic problem may not deal appropriately with the uncertainties. Stochastic programming and robust optimization have been used to tackle this type of problem. By considering different future scenarios, stochastic programming models can provide better solutions to handle future uncertainties well.

This thesis presents a stochastic ship routing problem with inventory management under uncertain consumer demands. We consider a single commodity distribution by a fleet of ships with limited storage at each source and destination port. The problem involves finding the least cost schedule for the fleet, and managing the inventory at each port. Different solution approaches are studied in order to solve the problem. The problem is firstly formulated as a straightforward MIP model. A set of integer variables are used to control the ship's routes and a group of continuous variables are used to manage the inventory situations at each port, as well as the capacity on board each ship. The MIP model is written in AMPL and solved directly by calling CPLEX. In the model, the status of each ship including the ship's position information and sailing information are decided by two integer variables. The storage level for each period is monitored by the inventory constraints. This method can however solve small problem only.

In order to solve larger problems within a reasonable time, the branch-and-price algorithm is developed. This algorithm finds the integer solution using branch-and-bound, and in each branch-and-bound node, it uses a column generation process to solve a restricted master problem. The master problem has the form of a set partitioning but with extra inventory constraints. Each column in the master problem corresponds to a tree of schedules for one ship, including the visit sequence, the start service time and the loading or unloading quantity. The problem at each branch-and-bound node is solved by the column generation method, which only requires a finite number of columns, and the subproblem for each ship is formulated as a shortest tree problem with time windows. The most promising columns for the master problem are generated by a stochastic dynamic programming using a backward labeling method. At each stage of the solution, the master problem is solved and the dual variables of the constraints are used within the stochastic DP to find the most negative reduced cost for each ship in the subproblem – there are the most promising columns for the master. The process is stopped if no more negative reduced cost columns can be generated, and a branch-and-bound algorithm is used to find the optimal integer solution.

In real life situations, ships can change their speeds or divert to other destinations during sailing. In order to implement these cases, extensions to the original decomposition model are developed. In the original model each ship is assumed to travel at a fixed speed. However, in practice a ship uses less fuel if it slows down the speed and so we extend the original model to allow for this. In the variable speed model, the edge cost structure has been modified in the dynamic programming networks of the subproblems to allow ships to choose different speeds during sailing. Because of the discretization of traveling time used in the model, a local optimization model is then proposed to determine the optimal schedules.

The case of diverting occurs in the real world because of a lack of future information, and when it occurs it results in higher traveling costs. It is often cheaper for the ship to sail to a central position in the sea and when future information becomes available to decide its destination. In terms of the stochastic ship routing problem, a divert model is built to capture this situation. The difficulty involved in a divert model is to find the suitable central position in the sea for diverting, and two ways are presented to find this central position for diverting, namely, a grid network and a pre-optimization method. The first method involves the use of a grid network and a set of discretised points in the network to find possible central points in the ship subproblems. A new type of sum-up node where position of central points is the state parameter is introduced in the DP network of subproblems, but there is no change made to the master problem, so the extension model can be solved by the same decomposition approach which is used to solve the original model. Each of these nodes has a fixed time and a cost function over the different possible central points. The second method considers all of

the possible situations which could occur when crossing the boundary of time period, and a set of nonlinear programming problems are solved a priori to generate the optimal central points for these situations. A list of central points for diverting and the corresponding costs of each situation is generated before solving the problem. When solving the DP, the cost function can be updated at each node by simply referring to this list of central points and costs. This way is compatible with the solution approach of the original model as well.

The thesis also presents an extension to the model allowing more general and realistic situations to be modeled. The model extension allows for the situation where a ship can take an extra work from an outside company and so usefully use time when it would otherwise be idle. The more realistic situations considered in the thesis are good complements to our original model.

The aims of this thesis are as follows:

- to describe the stochastic ship routing problem with inventory management,
- to study several solution approaches, both MIP and decomposition formulation, to solve this problem,
- to propose a range of extension models to the original model so as to capture some features involved in the real world shipping operations.

And the main contribution of this thesis are:

- In this thesis, a new MIP model for the stochastic ship routing problem is built. Balance equations and integer variables are used to control ship position and traveling status. And continuous variables are used to monitor the inventory level at each port.
- As far as we know, this is the first time that a Branch-and-Price algorithm is used to solve the stochastic ship routing problem. There is no paper where a similar method has been used to solve the stochastic transportation problem. In our solution framework, we formulate the ship schedule with scenario tree structure into a single column so that a set partitioning model can be used as our mater problem in the column generation method. Medium sized problems can be solved within reasonable time by using this decomposition formulation.
- In the column generation method, a new labeling method is, first time, introduced to solve the stochastic dynamic programming with time windows.
- When solving the stochastic DP, we eliminate 2-cycles during the labeling method. This is the first time that the details of 2-cycle elimination during labeling method is fully described.

- Two extension models, diverting and variable speed model, are introduced in the thesis to capture more realistic situations in ship operations. This is the first time that diverting and variable speed case are considered in the stochastic ship routing problem, and efficient solution methods with decomposition formulation are also presented.

The later chapters of the thesis have the following content, Chapter 2 gives a review of the existing literature in terms of the different routing problems. Chapter 3 describes the basic concepts of the problem. Chapter 4 give a mixed integer programming model for the problem. Chapter 5 gives a decomposition solution approach, together with a detailed solution framework, mathematical formulations, solving methods, branch-and-bound algorithms and computational results. Chapter 6 gives extension models, including the variable speed and diverting model, and finally, Chapter 7 presents the conclusion and a list of possible future work in this area.

Chapter 2

Literature Review

2.1 Traveling Salesman Problem with Time Windows

The traveling salesman problem (TSP) has received much attention throughout the literature because it is easy to describe, widely be applied to many transportation and distribution problems (such as vehicle routing, ship routing, crew scheduling and production planning), and difficult to be solved. In this section, we discuss different formulation of the TSP and different solution approaches given in the literature.

Consider a complete directed or non-directed graph $G = (N, A)$, where N is a set of n vertices and A is a set of m arcs. Let c_{ij} be the cost of arc (i, j). A Hamiltonian cycle of G is a circuit passing through every node of G once and only once. TSP is the problem of finding the Hamiltonian cycle in graph G with the least traveling cost. The mathematical formulations of the TSP given by Christofields et al. 1979 can be presented as blew.

Let binary variable $x_{ij} = 1$ if arc (i, j) is in the optimal TSP tour (Hamiltonian cycle), and 0 otherwise. The problem is then:

$$\min \quad \sum_{i=1..n} \sum_{j=1..n} c_{ij} x_{ij} \tag{2.1}$$

$$s.t. \quad \sum_{i=1..n} x_{ij} = 1, \qquad \forall j \in N \tag{2.2}$$

$$\sum_{j=1..n} x_{ij} = 1, \qquad \forall i \in N \tag{2.3}$$

$$\sum_{i \in K} \sum_{j \in K} x_{ij} \leq |K| - 1, \quad \forall K \subset N \tag{2.4}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \in N \tag{2.5}$$

In above formulation, K represents any nonempty subset of N and $|K|$ gives the size of set K. Constraints 2.4 ensure that x_{ij} $(\forall (i,j) \in A)$ must form a tour.

A special case of TSP is the symmetric traveling salesman problem. In the symmetric traveling salesman problem, graph G is a non-directed graph. Let l be the index of arc set A and c_l be the corresponding cost on the arc. Variable $x_l = 1$ if arc l is in the optimal TSP tour, and 0 otherwise. Let set $J(i)$ be the set of all non-directed edges linked with node i in G and $A(K)$ be the subset of A linking with the nodes in K (the nonempty subset of N). The symmetric TSP can therefore be formulated as:

$$\min \quad \sum_{i=1..n} \sum_{l \in J(i)} c_l x_l \tag{2.6}$$

$$s.t. \quad \sum_{l \in J(i)} x_l = 2, \qquad \forall i \in N \tag{2.7}$$

$$\sum_{l \in A(K)} x_l \leq |K| - 1, \quad \forall K \subset N \tag{2.8}$$

$$x_l \in \{0,1\}, \qquad \forall l \in A \tag{2.9}$$

Another special case for TSP is the traveling salesman problem with time windows (TSPTW). In TSPTW, again we have a graph $G = (N, A)$. Each node in G has a time window $[A_i, B_i]$. There is a depot in the network, and each arc (i, j) of the graph associates a traveling cost c_{ij} and a traveling time T_{ij}. The problem is to find the least cost Hamiltonian cycle covering each node (except the depot) exactly once within its time window. The tour starts and ends at the depot node. The formulation of TSPTW is given below:

$$\min \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2.10}$$

$$s.t. \quad \sum_{i \in N} x_{ij} = 1, \qquad \forall j \in N \tag{2.11}$$

$$\sum_{j \in N} x_{ij} = 1, \qquad \forall i \in N \tag{2.12}$$

$$x_{ij}(t_i + T_{ij} - t_j) \leq 0 \qquad \forall (i,j) \in I \tag{2.13}$$

$$A_i \leq t_i \leq B_i \qquad \forall i \in N \tag{2.14}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{2.15}$$

In the formulation, I is the set of arcs in the network except all the arcs linked with the depot node. Interval $[A_i, B_i]$ is the time window for node i. Constraints 2.13 ensure the elimination of subtours.

TSPTW is well-known to be an NP-hard problem. Even finding a feasible solution for TSPTW is an NP-complete problem (see Savelsbergh 1985).

Baker 1983 has presented a time-constrained TSP which minimized the total completion time and was solved by a branch-and-bound algorithm. The absolute value constraints and the time window constraints in the model are relaxed, and the dual of the relaxed problem is a longest path problem. In the branch-and-bound node, absolute value constraints are added to the model as the additional columns in the dual, which are the additional directed arcs in the dual network. The paper reports that the algorithm performed well on problems with up to 50 nodes with small percentage of the time windows overlap. Pesant et al. 1996 presented a constraint logic programming model for the TSPTW. Branch-and-bound algorithm is used in their paper to solve the problem. In their solution approach, they relax the time window constraints during the branch-and-bound. Langevin et al. 1993 have given a two-commodity flow formulation of the problem and solved the problem by a branch-and-cut algorithm.

Dumas et al. 1995 gave the new elimination tests to reduce the state space and use the tests to minimize the total traveling costs for TSPTW. They report that the new algorithm performs very well with dynamic programming approach. By using the time windows of nodes in the network, their three elimination tests eliminate states which cannot extend a partial path. An exact algorithm based on dynamic programming has been described in Mingozzi et al. 1997. Bounding functions are used in their paper to reduce the state space and these functions are based on the technique of generalization of the "State Space Relaxation" for dynamic programming which has also been introduced by Christofields et al. 981a.

Because of the difficulties in solving TSPTW, a lot of effort has gone on solving the problem heuristically. Gendreau et al. 1998 have described a generalized insertion heuristic algorithm for TSPTW. They first insert vertex into partial route to construct routes and then a post-optimization phase is used to improve the quality of the solution. This is proved to be a very successful way for large scale realistic problems. They reported the method performs well on 375 instances. Calvo 2000 has presented another insertion heuristic algorithm which includes two phases as well. For a relaxation of original problem, Greedy insertion is first used based on the solution of assignment problem. Then the solution is improved by the local search.

2.2 Vehicle Routing Problem

There are a set of customers and a depot in the vehicle routing problem (VRP). The problem is to determine K vehicle routes, one for each vehicle so that the total traveling cost is minimized. A route here is a tour that begins and ends at the depot, and visits a subset of the customers. There is a demand

for each customer in the problem, and each vehicle has a limited capacity on board. Each customer must be covered by exactly one of the K vehicle routes and total amount of demands for customers assigned to each vehicle must not exceed the vehicle capacity.

There are two types of problems in the VRP, Capacitated VRP (CVRP) and VRP with Time Windows (VRPTW). VRPTW is the extension of the CVRP in which capacity constraints are imposed and each customer i is associated with a time window $[A_i, B_i]$, called a time window. The customer has to be visited or serviced with the time window. VRPTW is regarded as NP-hard problem.

2.2.1 Vehicle Routing Problem with Time Windows

In this section, we focus on the VRPTW. We first describe the mathematical formulation of VRPTW, and then we review the different solving approaches in the literature in later sections. Let $G = (N, A)$ be a graph, where $N = \{0, .., n\}$ is the vertex set and A is the arc set. Here vertex $1, .., n$ correspond to the customers and 0 corresponds the depot. Each customer has a demand d_i which should be satisfied by vehicle visiting. There are a traveling cost, C_{ij}, and a traveling time, T_{ij}, on each arc $(i, j) \in A$. In each customer i, a time window $[A_i, B_i]$ is associated with it and the service to the customer should start within the time window. K vehicles are considered in the problem. The VRPTW consists of finding exact K routes, one for each vehicle, with minimum traveling cost and satisfying the following constraints:

- Each route must start and end at the depot;
- Each customer vertex must be covered by exactly one route;
- The sum of the demands of vertices visited by a vehicle can not exceed the vehicle capacity, C;
- For each customer i, the service start time must be within the corresponding time window.

Assume that flow variables x_{ijk} is 1 if arc (i, j) is used by vehicle k and 0 otherwise. Variable t_{ik} is the start of service at node i when serviced by vehicle k. Then the formulation of the VRPTW can be presented as below.

$$\min \quad \sum_{k\in K}\sum_{(i,j)\in A} C_{ij}x_{ijk} \tag{2.16}$$

$$s.t. \quad \sum_{k\in K}\sum_{j\in N} x_{ijk} = 1 \qquad \forall i \in N \setminus \{0\} \tag{2.17}$$

$$\sum_{j\in N} x_{0jk} = 1 \qquad \forall k \in K \tag{2.18}$$

$$\sum_{i\in N} x_{ijk} - \sum_{iinN} x_{jik} = 0 \qquad \forall k \in K, j \in N \tag{2.19}$$

$$\sum_{i\in N} x_{i0k} = 1 \qquad \forall k \in K \tag{2.20}$$

$$x_{ijk}(t_{ik} + s_i + T_{ij} - t_{jk}) \leq 0 \qquad \forall k \in K, (i,j) \in A \tag{2.21}$$

$$A_i \sum_{j\in N} x_{ijk} \leq t_{ik} \leq B_i \sum_{j\in N} x_{ijk} \qquad \forall k \in K, i \in N \tag{2.22}$$

$$\sum_{i\in N} d_i \sum_{j\in N} x_{ijk} \leq C \qquad \forall k \in K \tag{2.23}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in A \tag{2.24}$$

The objective function 2.16 is to minimize the total traveling cost. Constraints 2.17 ensure that each customer is visited by one vehicle. Constraints 2.18 and 2.20 describe that a vehicle route must start and end at the depot node 0, while constraints 2.19 shows the flow balance equation on the route followed by vehicle k. The nonlinear constraints 2.21 and constraints 2.22 ensure each node in a vehicle route should be serviced within the corresponding time window if an arc is included in the vehicle's route. Constraints 2.23 guarantee the total demand delivery does not exceed the capacity of vehicle on the route of each vehicle. Finally, constraints 2.24 show x_{ijk} are binary variables.

The formulation is a nonlinear programming problem. However, by using the binary conditions of variable x_{ijk} and a large constant M, the constraints 2.21 can be linearized as:

$$t_{ik} + s_i + T_{ij} - t_{jk} \leq (1 - x_{ijk})M \qquad \forall k \in K, (i,j) \in A \tag{2.25}$$

2.2.2 Decomposition Approaches – Column Generation

The column generation approach represents a generalization of the Dantzig-Wolfe decomposition described in Dantzig and Wolfe 1960. In the approach, the problem is divided into two parts: master problem and subproblems. The master problem can be formulated as a set partitioning model, and

subproblems are usually formulated as a shortest path problem with time windows (SPPTW) which is solved by dynamic programming. In the formulation of VRPTW introduced above, the master problem contains the objective 2.16, and constraints 2.17 and 2.24. All other constraints are considered in the subproblems.

Let Ω be the set of paths, and path $p \in \Omega$ be a possible path generated by subproblems. Each path p then can be described using integer flow value $\hat{x}_{ijkp}$. Let index k be vehicle. For a given $k \in K$, we have:

$$x_{ijk} = \sum_{p \in \Omega} \hat{x}_{ijkp} \theta_{kp} \quad \forall (i,j) \in A \tag{2.26}$$

$$\sum_{p \in \Omega} \theta_{kp} = 1, \ \theta_{kp} \geq 0 \qquad p \in \Omega \tag{2.27}$$

Now the master problem can be formulated as a set partitioning model:

$$\min \quad \sum_{k \in K} \sum_{p \in \Omega} C_{kp} \theta_{kp} \tag{2.28}$$

$$s.t. \quad \sum_{k \in K} \sum_{p \in \Omega} a_{ikp} \theta_{kp} = 1 \qquad \forall i \in N \tag{2.29}$$

$$\sum_{p \in \Omega} \theta_{kp} = 1 \qquad \forall k \in K \tag{2.30}$$

$$\theta_{kp} \geq 0 \qquad \forall k \in K, p \in \Omega \tag{2.31}$$

$$x_{ijk} = \sum_{p \in \Omega} \hat{x}_{ijkp} \theta_{kp} \qquad \forall k \in K, (i,j) \in A \tag{2.32}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in A \tag{2.33}$$

From above formulation, we can see that each column in the set partitioning model represent a route for a vehicle. In practice we cannot a priori find all the possible routes for each vehicle to add into the set partitioning model. Instead we repeatedly solve a subproblem to generate promising columns. In the Dantzig-Wolfe decomposition approach, the master problem is solved to provide the dual variable of constraints 2.29 and 2.30. Subproblems then use this dual variable information to generate the least reduced cost routes. The routes are added into the master problem as columns if the corresponding reduced costs are negative. The process is stopped if no more negative reduced cost route can be generated from the subproblems.

An early paper, Appelgren 1969, Dantzig-Wolfe decomposition is used for a ship scheduling problem, which is a similar problem to VRPTW. They show that the frequency of fractional solutions is about 1-2 percent for their test examples. Problems with about 40 ships and 50 cargoes were solved in about

2.5 minutes.

The Dantzig-Wolfe decomposition approach has been successfully applied to solve the VRPTW in Desrochers et al. 1992. The LP relaxation of the set partitioning formulation of the VRPTW is solved by a column generation method in the paper. Their master problem is a set covering model and the subproblems are formulated as dynamic programming problem. Three dynamic programming models are discussed. One is able to price out all feasible routes while the other two have the solution spaces containing more than the feasible route set but having a pseudopolynomial worst case complexity. Finally the branch-and-bound algorithm is discussed in their paper.

Based on the same set partitioning model given in Desrochers et al. 1992, Gelinas et al. 1995 have presented new branching strategies, branching on resource variables rather than branching on flow variables. They do branching on time windows to get more balanced branch-and-bound tree. The big time window on a selected node of network is divided into two small time intervals, one in each branch. The backhauling was permitted only after all the shipments to clients have been made in the model. They report that the new branching method is very effective, and they can optimally solve problems with up to 100 customers.

When the VRPTW is solved by column generation, the optimal solution value of the set covering problem is usually very close to its linear programming relaxation, which makes the branch-and-bound algorithm very efficient. Bramel and Simchi-Levi 1997 explained this behaviour. they show that for any distribution of service times, time windows, customer loads and locations, the gap between fractional and integer solutions of the set covering problem becomes very small with the number of customers increased.

A multi-depot vehicle scheduling problems with time windows and waiting costs is presented in Desaulniers et al. 1998. The problem with waiting costs provides a more realistic cost structure. They formulate the problem as an integer nonlinear multi-commodity network flow model and solve the problem using a column generation approach and integer solution is searched by branch-and-bound algorithm. They tested their model on randomly generated examples and found that such a general solution methodology can efficiently solve the case with exact waiting costs.

More recently, Chabrier 2006 proposes one theoretical and several practical improvements to the algorithm which uses the elementary shortest path subproblems to generate elementary paths for the master problem.

2.2.3 Decomposition Approaches – Lagrangian Relaxation

Another useful decomposition approach for solving the VRPTW is Lagrangian relaxation. By relaxing the time and capacity constraints, 2.21 and 2.23 respectively, the relaxed problem becomes just a network flow problem. However, integer solution is rarely to be found by doing so because that the integrality gap is very large. Therefore, the time and capacity constraints should be kept in the problem and instead some network flow constraints can be relaxed. This improves the lower bound of the Lagrangian relaxation.

Kohl and Madsen 1997 presents an optimization algorithm for the VRPTW based on Lagrangian relaxation. They relax the constraints 2.17 in the formulation of VRPTW, which models the fact that each customer must be served. Therefore, the objective for their Lagrangian relaxation problem has the following formulation:

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} C_{ij} x_{ijk} + \sum_{k \in K} \sum_{i \in N} \alpha_i (1 - \sum_{j \in N} x_{ijk}) \tag{2.34}$$

They propose subgradient and bundle methods to find the optimal multiplier values in the master problem, and their subproblem is a shortest path problem with time windows and capacity constraints. Their algorithm is shown to be very competitive compared to other algorithms after testing a series of well-known benchmark problems of size up to 100 customers.

Fisher 1994 extends the 1-tree method to a K-tree method for the VRPTW. The capacity constraints and the requirement that each customer be visited exactly once are relaxed. In their method, it is assumed that a route containing only one customer is not allowed. Feasible solutions are obtained at each subgradient iteration using three alternative heuristics based on the Lagrangian solution. They can optimally solve a number of difficult problems and several real problems with 25–71 customers. A later paper, Fisher et al. 1997, describes two optimization methods for VRPTW, Lagrangian relaxation and K-tree approach.

2.2.4 Heuristic Approaches

In the real world, the problem size is usually very big, so it is very important to get a good qualty sub-optimal solution within a reasonable time rather than take a long time to find the optimal solution. Therefore, heuristic approaches for solving the VRPTW can play a very important role. There is a large amount of literature solving VRP by heuristic methods. Meta-heuristics are proved to be very efficient heuristic methods to get good quality solutions. For VRPTW, solution is usually found by heuristically constructing feasible

routes and improving the routes by local searches.

Solomon 1987 has considered a group of heuristic methods for route construction. They insert node into a partial route based on the measure of traveling time and cost. Their computational results focus on the insertion heuristic. The method used two criteria at every iteration to insert an un-routed customer into the current partial route between two adjacent customers. They first found the best feasible insertion place for each un-routed customer by computing the minimum additional distance or time required, and then they found the best customer to be inserted into the place. Based on the k-interchange concept, Savelsbergh 1985 has discussed a local search approach in routing problems with time windows. A k-interchange finds the feasible routes by swapping k links of a route with k other links. In their work, they considered 2-interchange and 3-interchange. This paper showed an alternative way of building routes from existing routes which is different from the method of route construction given in Solomon 1987.

Unlike local search heuristics, meta-heuristics do not stop at the local optimal solution. Tabu search heuristics are based on the local search which requires the feasibility of solution all the time, and the recent changes will be forbidden to be recovered within a certain steps. Potvin et al. 1996 have described a tabu search heuristic for the VRPTW. Both 2-opt* and Or-opt neighborhoods were used to find good solutions. They introduced distance restrictions so as to avoid to search the entire 2-opt* and Or-opt neighborhoods. A number of examples were tested with different parameter settings. Taillard et al. 1997 presented a tabu search heuristic for the vehicle routing problem with soft time windows with a penalty if services to customers are out of the time windows. In a neighborhood of the current solution, new solutions were searched for using the CROSS exchange. The routes of the best previously visited solutions were stored in memory, and new starting points for the tabu search were produced by a combination of routes taken from different solutions stored in the memory. According to Toth and Vigo 2002, the CROSS exchange generalizes both the 2-opt* (Potvin and Rousseau 1995) and Or-opt exchanges, but it is a special case of the k-interchanges (Osman 1993) since it has the restriction that the subsets of customers chosen in each route must be consecutive.

Tabu search is also used to search for good solutions to different extensions of the VRPTW. Cordeau et al. 1997 have presented a tabu search heuristic for the periodic vehicle routing problem (PVRP) and the multi-depot vehicle routing problem (MDVRP). Their tabu search algorithm was based on the GENI heuristic. Semet and Taillard 1993 have described a real-life vehicle routing problems, which is quite different from the academic problems in terms of problem size and number of restrictions, solved by tabu search. Therefore, the real-life problems are very hard to solve to optimality, and heuristic methods play an important role in practice. In the paper, they dealt with varieties of restrictions from real-life problems and built suitable

heuristics to solve the problem efficiently.

There is as well a large amount of literature using other heuristic approaches. Metastrategy simulated annealing algorithms were presented in Osman 1993 and Chiang and Russell 1996. A revolutionary heuristic algorithm, genetic search, was proposed in Potvin and Bengio 1996 as well as Baker and Ayechew 2003. In Baker and Ayechew 2003, both a pure genetic algorithm and a hybrid of the genetic algorithm with a neighborhood search method were tested, and the later hybrid method was competitive with tabu search and simulated annealing in terms of solution time and quality. Russell 1995 has presented a hybrid heuristic algorithm which embeds global tour improvement procedures within the route construction process for the VRPTW. A location based heuristic has been introduced in Bramel and Simche-LEVI 1995 for general routing problems including VRP. Another paper dealing with the soft time windows on the VRP is Balakrishnan 1993. They have described three simple heuristics for the problem.

2.2.5 Vehicle Routing Problem with Uncertainties

A variant of the VRP is the stochastic vehicle routing problem (SVRP). The uncertainties involved in the VRP are the locations of customers and the customer demands. These uncertainties can be dealt with by stochastic programming and the robust optimization, and can be solved by both exact optimization methods and heuristics. The objective of the SVRP is to minimize the expected route failure, maximize the customer coverage, or minimize the total expected traveling cost (or distance).

Recently, stochastic vehicle routing problem without inventory constraints and with simple recourse actions have been discussed extensively in the literature. A branch-and-price algorithm for vehicle routing problem with stochastic demands is illustrated by Christiansen and Lysgaard 2007. 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.

Bent and Hentenryck 2004 have considered a dynamic VRPTW with stochastic customers. The number of served customers was maximized. A multiple scenario approach (MSA) that continuously generates routing plans for scenarios including known and future requests was presented. They tested their model on the Solomon benchmark problems with a degree of dynamism varying between 30–80 percent.

A straight-forward modification of the Clark and Wright savings algorithm for the SVRP was given in Dror and Trudeau 1986 based on a discussion of route failure. In Hjorring and Holt 1999 and Gendreau et al. 1995, an integer L-shaped method was used to solve SVRP to optimality. Bertsimas and Simchi-Levi survey some new developments in SVRP in Bertsimas and Simchi-Levi 1996. Bertsimas 1992 proposed a priori sequence among all customers of

minimal expected total length, and a variety of theoretical approaches were analyzed as well. In addition, several solution frameworks for the stochastic vehicle routing with stochastic demands were discussed in Dror et al. 1989.

Heuristic approaches are also used for SVRP with stochastic demands and customers. Gendreau et al. 1996 presented a tabu search heuristic for the VRP with uncertain demands and customers. A genetic based algorithm was proposed in Guo and Mak 2004. Soft time windows with a penalty for late arrival were used in their paper. A mathematical programming formulation was developed to study the effects of the stochastic demands and customers.

2.3 Ship Routing Problem

Ocean shipping is the major international transportation mode especially for large bulk commodity transportation. Ocean transportation involves shipping products among ports with very large distances from each other, which is one of main characteristics of ocean shipping industry. An example is the international oil shipping operations between continents. Another characteristic of ocean shipping is the large volumes on board. Alternatively, the sea shipping may be within a relatively small area, such as the ferryboat service among the small Greek islands in the Aegean Sea.

The ship routing problem (SRP) is part of the fleet planning problem for ocean shipping. Ship routing and scheduling problems are different from those of other transportation problems. Ronen 2002 has pointed out some significant differences:

- In ship routing problem, there may be different types of commodities needed to be transported in a single vessel.
- Ship routing problem always consider the inventory at the sources and customers.
- A voyage of a vessel usually has one or very few unloading locations with the result that ships may travel frequently between sources ports and consumer ports. In vehicle routing problem in contrast, a vehicle may visit a long sequence of customers in a journey.
- Traveling times are usually much longer than other transportation planning problems (days or weeks instead of hours).
- (Optional) backhauls may be available for the vessels.
- Vessels come in a large variety of sizes with different compartment sizes.

In addition to the differences mentioned above, there are some other differences between SRP and others. Ships may stay at a port for a long time

compared to the VRP because the large capacity on board results in longer loading or unloading times. Diverting to a new location is not considered in the VRP, but ships divert during sailing in real life. Christiansen et al. 2004 have also provided some additional differences.

There is an extensive literature on deterministic inventory routing in the content of ship routing problem. 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 their paper.

Different solution methods are presented in the literature. 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. In their problem, 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 an 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 directly. Heuristic method then is used to solve this real industry problem. 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 mixed integer programming model.

The Dantzig-Wolfe decomposition approach has proved to be successful for the vehicle routing problem with time windows. For the SRP, it is also a good solution approach. There is much literature on solving SRP 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.

Christiansen 1999 describes a real world ship routing problem. The MIP model is in their work. To decompose the MIP model, the model is reformulated into a set partitioning model and solved by a Dantzig-Wolfe decomposition approach. The same problem is also given in Christiansen and Nygreen 1998a. They introduce a master problem with coupling constraints as below:

$$\min \quad \sum_{v}\sum_{r} C_{vr}\lambda_{vr} \tag{2.35}$$

$$s.t. \quad \sum_{v}\sum_{r} A_{imvr}\lambda_{vr} + \sum_{s} Y_{ims}\theta_{is} = 1 \qquad \forall i, m \tag{2.36}$$

$$\sum_{v}\sum_{r} Q_{Vimvr}\lambda_{vr} - \sum_{s} Q_{Hims}\theta_{is} = 0 \qquad \forall i \in S_H, m \tag{2.37}$$

$$\sum_{v}\sum_{r} T_{Vimvr}\lambda_{vr} - \sum_{s} T_{Hims}\theta_{is} = 0 \qquad \forall i \in S_H, m \tag{2.38}$$

$$\sum_{r} \lambda_{vr} = 1 \qquad \forall v \tag{2.39}$$

$$\sum_{s} \theta_{is} = 1 \qquad \forall i \in S_H \tag{2.40}$$

$$\lambda_{vr} \geq 0,\ \theta_{is} \geq 0 \tag{2.41}$$

$$\{r : \lambda_{vr} > 0\} \textit{correspond to equal geographical routes} \quad \forall v \tag{2.42}$$

Where variable λ_{vr} is 1 if ship v chooses route r and variable θ_{is} is 1 if port i chooses visit sequence s. Objective function 2.35 minimizes the traveling cost. Constraints 2.36 ensure that each port visit can occur at most once. Constraints 2.37 and 2.38 are used to couple the load quantity and start service time for the ship visit sequences and port visit sequences. Constraints 2.39 and 2.40 ensure that each ship must choose a visit sequence and each port must choose a visit sequence as well respectively. Constraints 2.42 are only implicitly given and represent the integer part of the model. Detail explain of the master problem is given in Christiansen and Nygreen 1998a. Columns were generated from ship and port subproblems, and depth first branch-and-bound algorithm was used to search for the integer solution. In Christiansen and Nygreen 1998b, the same authors focus on the formulation of subproblems. Ship subproblems are formulated as the shortest path problem with time windows and solved to generate the ship visit sequence including start service times and load quantities, while port subproblems are used to generate port visit sequences for the master problem. Both subproblems were solved by dynamic programming. More details about the solution approach have been given in Christiansen 1996.

Instead of generating columns iteratively, generating good columns for set partitioning model a priori is another solution approach for ship routing problem. A good heuristic or exact method for getting promising columns before solving the set partitioning model is important for getting good quality solutions within a short solving time. Bronmo et al. 2006 present a set partitioning approach to solving the multi-ship routing with time windows and flexible cargo sizes. The columns in the model are generated a priori and the best ship schedules are chosen by solving the model. Fagerholt

and Christiansen 2000a present a bulk ship scheduling problem that is a combined multi-ship pickup and delivery problem with time windows and multi-allocation problem. Each ship can be divided into several compartments of different sizes by using bulkheads. A two-phase set partitioning approach is presented to solve the problem. In Phase one, feasible schedules are generated for each ship by a heuristic insertion method, and the optimal visiting sequence based on the current obtained schedules are found by solving a Traveling Salesman Problem with Allocation, Time Windows and Precedence Constraints (TSP-ATWPC). Then in phase two, a set partitioning model is solved to choose the best sequence from generated sequence candidates. The method for solving TSP-ATWPC is described in Fagerholt and Christiansen 2000b.

Fagerholt 1999 describes a real liner shipping problem. A three-phase solution approach is developed in their work. In phase 1, all feasible single routes are generated for the largest ship available. The route generated in phase is divided into multiple routes by smaller size ship. Then by solving a set partitioning problem based on the routes generated in phases 1 and 2, the optimal solution is found. A ship routing problem with soft time windows is described in Fagerholt 2001. It is implemented by introducing penalty costs for when the service is out of the customer time windows. A set partitioning model is introduced to solve the problem. All the columns are generated a priori. The physical routes are first enumerated and the detailed schedules with service information based on the physical routes are found by solving an optimization problem. Bausch et al. 1998 introduce a short-term marine transport of bulk products problem. Their columns for the set partitioning model are generated a priori by a highly detailed simulation inside a spreadsheet.

Many extensions of the ship routing problem and different solution approaches are also discussed in literature. Xie et al. 2000 propose an integer programming model combined with dynamic programming improvement process for the fleet planning problem. In their model, a number of new ships can be added to the fleet within the planning horizon. An improved formulation of bulk cargo ship scheduling (stated in Ronen 1986) is given in Cho and Perakis 2001. They linearize the nonlinear part of the original model and reformulated the model as a linear integer programming model with fewer number of integer variables. Fagerholt 2004 presents a total different approach based on a optimization-based decision support systems (DSS).

There are few references to work on the stochastic inventory routing problems concerning ships. Christiansen and Fagerholt 2002 is one paper in the area. It presents robust ship scheduling with multiple time windows. A more realistic situation of time windows is considered in their model. They consider that ports are closed for service at night and during weekends, the wide time windows can be split into multiple time windows. A set partitioning approach with the columns found a priori is proposed. The method aims 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. Azaron and Kianfar 2003 apply stochastic dynamic programming to find the dynamic shortest path from the source node to the sink node in stochastic dynamic networks. This is another work involved with a continuous time Markov process.

2.4 Methods for Solving Shortest Path Problem with Time Windows

As discussed above, in the column generation decomposition approach and the Lagrangian relaxation decomposition approach for both vehicle routing and ship routing problem, the subproblems are usually formulated as shortest path problems with time windows. Therefore, finding efficient methods to solve the SPPTW quickly has gained a lot of attention in literature.

The SPPTW consists of finding the least cost route between a source o and a sink d in a network $G = (N, A)$. Each node $i \in N$ associated time window $[A_i, B_i]$, and the node has to be visited with its time window. There is a cost $\bar{C}_{ij}$ and a strictly positive duration (or traveling time) T_{ij} for each arc $(i, j) \in A$. Clearly, all arcs (i, j) respect the condition:

$$A_i + T_{ij} \leq B_j \tag{2.43}$$

Suppose variable x_{ij} is a binary flow variable which is 1 if the arc (i, j) is in the shortest path and 0 otherwise. t_i is a time variable which represents the start service time at node i. The SPPTW can be formulated as following optimization model:

$$\min \quad \sum_{(i,j)\in A} \bar{C}_{ij} x_{ij} \tag{2.44}$$

$$s.t. \quad \sum_{j\in N\setminus\{o,d\}} x_{oj} = 1 \tag{2.45}$$

$$\sum_{i\in N\setminus\{o,d\}} x_{id} = 1 \tag{2.46}$$

$$\sum_{j\in N\setminus\{o,d\}} x_{ij} - \sum_{j\in N\setminus\{o,d\}} x_{ji} = 0 \qquad \forall i \in N \setminus \{o,d\} \tag{2.47}$$

$$x_{ij}(t_i + T_{ij} - t_j) \leq 0 \qquad \forall (i,j) \in A \tag{2.48}$$

$$A_i \leq t_i \leq B_i \qquad \forall i \in \{o,d\} \tag{2.49}$$

$$A_i(\sum_{j:(i,j)\in A} x_{ij}) \leq t_i \leq B_i(\sum_{j:(i,j)\in A} x_{ij}) \quad \forall i \in N \setminus \{o,d\} \tag{2.50}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.51}$$

According to Desrochers and Soumis 1988a, the positive duration and time windows guarantee the finite number of feasible paths, but they do not guarantee the feasible paths will be elementary. Then, with each path x_{oj} from the origin o to the node j satisfying time windows, is associated a (time, cost) label corresponding to the arrival time at node j and the cost of the path x_{oj}, respectively. Denote (T_i^k, C_i^k) as the k^{th} path from o to i. The labels (T_i^k, C_i^k) can be calculated iteratively along the path $x_{pj} = (i(0), i(1), ..., i(L))$ as follows:

$$T_{i(0)} = 0 \tag{2.52}$$

$$C_{i(0)} = 0 \tag{2.53}$$

$$T_{i(l)} = \max\{a_{a(i)}, T_{i(l-1)} + d_{i(l-1)i(l)}\} \tag{2.54}$$

$$C_{i(l)} = C_{i(l-1)} + \bar{C}_{i(l-1)i(l)} \tag{2.55}$$

Based on the above concepts, Desrochers and Soumis 1988a have presented an efficient generalized permanent labeling algorithm for the SPPTW. A new definition of the concept of a generalized bucket was given, and a specific way for ordering the labels in the set of untreated labels was proposed as well. The specific order was then used to choose the next untreated labels to be handled each time. The positive duration of the arcs ensure that it is impossible to improve a label once it has been treated. With their generalized permanent labeling algorithms (GPLA), problems with up to 2500 nodes and 250000 arcs have been solved and their algorithm runs in pseudo-polynomial time. Desrochers and Soumis 1988b have presented a primal and dual re-optimization algorithm for the SPPTW. The cost of repeatedly solving the

SPPTW can be reduced by reusing part of the solution of the preceding problem.

Ioachim et al. 1994 have proposed a dynamic programming algorithm for the SPPTW with linear node costs. The node costs in their model are specified as a function of arrival time at each node in the network. This cost function of a path is piecewise linear and convex over a reduced time window and contains only a finite number of linear pieces. When updating the node cost function of each node in the network, suppose that q predecessors to the node are considered and several piecewise linear cost functions are obtained. These piecewise linear cost functions may partly dominate each other and the resulting cost function of the node is a piecewise linear, convex, might discontinuous and non-increasing function and contains at most q linear pieces. The algorithm performs better than the approaches of discretization of time windows, especially for the problems with wide time windows and many nodes with negative costs.

Chapter 3

Problem Description

There are four major transportation types in the current transport industry, namely, inland transport, air transport, pipeline transport and ocean transport. The inland transport industry is suitable for transporting cargo within the local area or the mainland of a continent. In a relatively small region, trucks or other vehicles can transport commodities efficiently and flexibly, and this procedure is inexpensive. In the mainland of a continent, trains can handle the transportation of a large amount of commodities cheaply, although the cost of building a railway may cause increased fees. Air transport is widely used for international business, and this is much more efficient than railway or marine transport, although it is not suitable for transporting large numbers of products because of the expensive charge. As its name suggests, pipeline transport is commonly used to transport bulk products, such as oil and liquefied natural gas, since this is suitable for ensuring a continuous supply of these products. However, in most cases, the supply countries are far away from the consumer countries. For example, it is not feasible to provide oil from Middle East countries to the United State via a pipeline, since it would be very expensive to build and maintain such a pipeline. Thus, in the operation of transporting huge amounts of products between international destinations, marine transport has a particular advantages compared to other transporting methods, it is cheap, flexible and has a huge capacity, and it is especially suitable for bulk commodities.

The model in this thesis is designed for the problem of transporting a single commodity. A good example which is distributed in a dedicated transport fleet is liquefied natural gas (LNG). Natural gas is liquefied in a continuously operating refrigeration plant, and the resulting LNG is fed into storage tanks. LNG is transported from these storage tanks by ship. The ships deliver the LNG to storage tanks near the consumer, and the LNG from these tanks is fed into a re-gassification plant to meet consumer demand for gas. This thesis focuses on general single bulk commodity shipping and storage operations, but not on any specified bulk product. However, after making necessary changes or assumptions, our model and solution approach can also be used for the

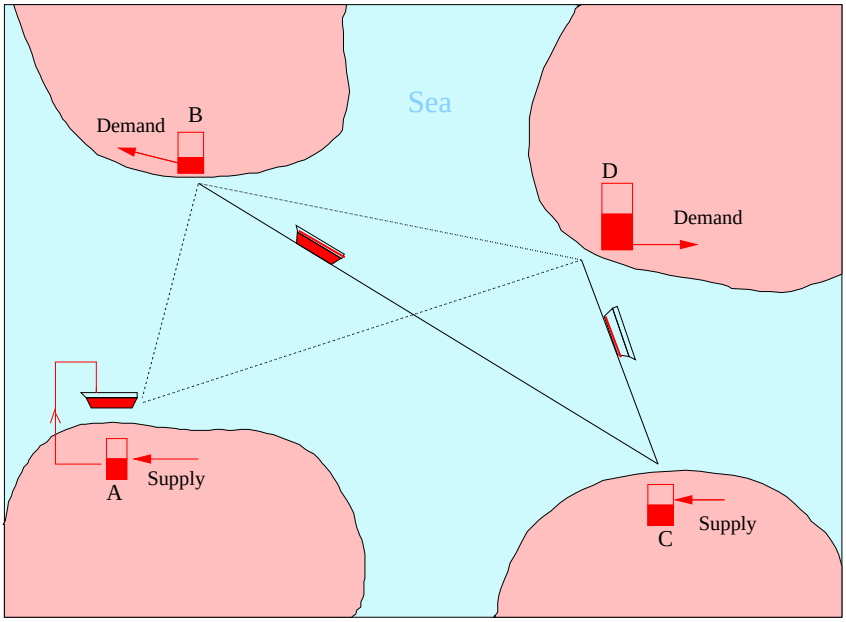


Figure 3.1: Transportation network

problem dealing with specified products.

The remainder of the chapter is devoted to giving some basic concepts of the model to help to understand the background of the problem. Transportation network is shown in Section 3.1, and Section 3.2 presents the objects of the problem. Detailed information about ships, demand, supply, schedule and storage are introduced in Section 3.3 – 3.6. The deterministic and stochastic versions of the model are compared in Section 3.7. Section 3.8 introduces the planning period, how to deal with end-of-horizon of the period, scenario tree, and the relationships among them, and the repeated deterministic method based on scenario tree is described in Section 3.9. Section 3.10 introduces the concept of the port visit.

3.1 Transportation Network

The ship routing problem normally contains a transportation network, including a set of ports (denoted by N) and a set of edges linked with different ports (denoted by E). There are two different types of ports in the network, namely consumer ports and supplier ports, and there is a finite storage capacity at each port. Each edge of the network is associated with a traveling time and a traveling cost (primarily a function of distance).

Figure 3.1 illustrates an example of a transportation network. There are four ports in the network, and ports A and C are suppliers, while ports B and D are consumers. A fleet of ships are sailing between the ports, loading at the supplier ports and discharging at the consumer ports. At each port,

there is a limited capacity tank to store the bulk product.

In the real situation, ships may not sail in a straight line between two ports. However in some later sections, we assume that ships sail in a straight line to simplify the problem. But our model does not require that ships must sail in a straight line. Without this assumption our model will still be valid.

3.2 Objects of the Problem

In the ship routing with inventory management problem, a heterogeneous fleet of ships is given for the planning period. The problem is to find the least cost schedules for the ships, which can also satisfy the supply and demand constraints. There are two main aspects of the problem. One is the need to manage the inventory levels at all times so as to keep the stock level within a suitable interval, not exceeding the upper limit of each stock, or going below the lowest allowed level at each port. The other aspect is the need to minimize the total traveling cost.

Different models may have different objectives for different purposes. Some models consider buying and selling products, as well as deciding ships' schedules, and the objective is to maximize the total profits (income minus operating costs). In other models, the schedule is designed for rolling periods, and ships are required to finish their transport tasks as early as possible. Hence the objective is to minimize the traveling cost as well as to minimize the finishing service time. The models introduced in later chapters of the thesis have the objective of minimizing the total traveling cost.

3.3 Ships

A fleet of ships is considered in the model, and each ship is allocated an optimal schedule after solving the problem. In the ocean transportation industry, each ship can be characterized by capacity, voyage cost and speed. In a realistic operation, a shipping company may own many ships of different sizes and speeds. Furthermore, the number of crew on each ship is also different from one ship to another so that the operating costs are varied among ships. However, it is assumed that each ship in this model has the same capacity, speed and traveling costs so as to simplify the problem. However, this is not a limitation of our model. In fact our model can handle the situation that the vessels are different from each other by using specified data.

The ships used for specified purposes have their own special characteristics. For example, LNG ships are normally special purpose vessels which are designed to move natural gas between continents in its liquid state. The LNG is stored in a special containment system within the inner hull, where it is kept at atmospheric pressure and cryogenic temperature, so as to keep the natural

gas in a liquid state. Keeping the pressure and temperature is expensive so that LNG ships cannot be fully empty when they arrive in the consumer port. A certain amount of LNG needs to be kept on board the ship otherwise it needs to be cooled down again, which is very expensive. This thesis will not deal with such specific LNG shipping issues, but only deals with the general characterisitcs of bulk transporting ships. However, the special characteristics for ships with specified purposes can be dealt with by giving extensions to the model built in the thesis by some necessary changes to the model.

3.4 Demand and Supply

Two of the most important factors involved in the model are the demand and supply rates. In the consumer port, the commodity is unloaded from ships and injected into storage, and it is taken out of storage continuously according to the demand. In the supplier port, the commodity is injected into storage after production, and loaded onto ships during the service. These two factors are very unpredictable in the real world. The rate of demand depends upon selling prices, market requirements, the global economic situation, and so on, while the rate of supply depends upon production rates, the market, and even politics. In order to handle the uncertainties of the demand and supply rate, the problem is needed to be built and solved by stochastic programming models. In the stochastic model presented in this thesis, the supply rate is assumed to be deterministic and the demand rate is the only uncertainty in the problem.

3.5 Ship Schedules

A ship's schedule will be specified by the route it follows, the ports it visits, the loading or unloading quantity at each port, and the start time of loading or unloading (which we refer to as service). Two schedules having the same physical visiting sequence but different service times or loading quantities, are regarded as being two different schedules. Only when they have the same physical routes, service times and loading quantities, are the two schedules regarded as being the same. Our model in the thesis is used to generate the physical visiting route of each ship and to decide the start service time, and the quantity loaded or discharged at each port of call. The structure of ship schedules for deterministic and stochastic model are different, and we discuss this later in Section 3.7.

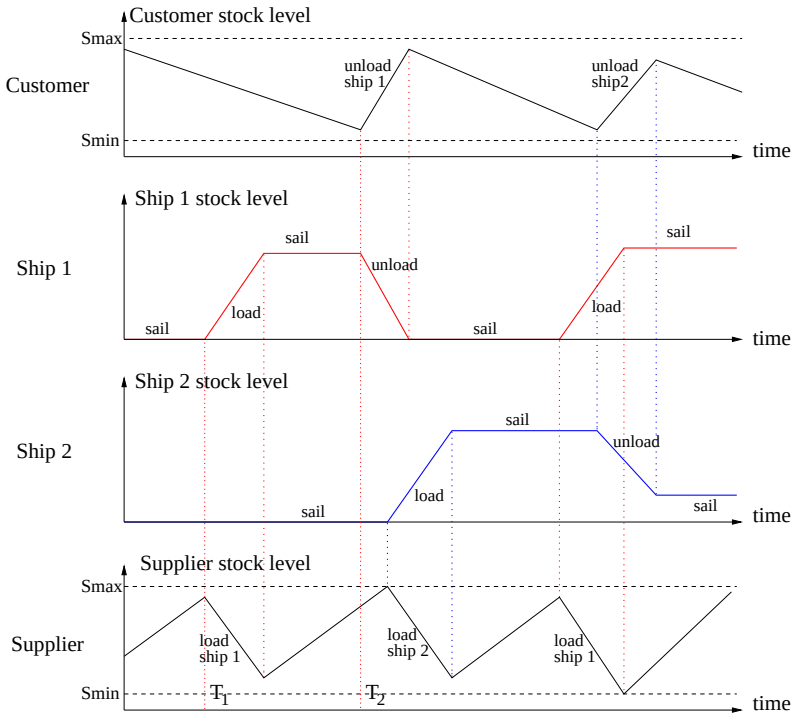


Figure 3.2: A small example of storage level

3.6 Storage

The inventories are considered at both consumer and supplier ports. The stock level of the consumer port storage goes down over time because the commodity is taken out of storage continuously as demand, unless a ship arrives and a further amount of the product is injected into the storage from the ship. The situation of the supplier port is just the opposite, and the stock level goes up because of the production rate and goes down if an amount of the commodity is loaded onto a ship.

The stock level needs to be recorded at the start and end of each service so as to be monitored during the planning period. Since the capacity for storage is finite, it has upper and lower bounds. The stock level of storage should be between the upper bound and lower bound all the time. The lower bound may not be 0. This is because that at a consumer port, it is usually a positive value in order to keep a safety level in each storage facility and avoid running out of stock.

The rates of change in storage levels are closely related to the demand or supply rate, and the ship's loading or unloading services. A simple example of two ships traveling between two ports, a supplier and a consumer, is shown in figure 3.2. This example illustrates the changes of stock level, the relationship between storage levels and ship services, and the amount of commodity changes on board the ships.

In our model we assume that within a time period, the rate of production and the rate of consumption are constant, and also the rate of transfer from storage onto a ship at a supplier port and the rate of transfer from a ship to storage at a consumer port are constant. But they can be different in different scenario node of the scenario tree. Consequently all the changes of level occur at a constant rate and so the level in the inventory is a continuous piecewise linear function of time. This behaviour can be seen in the example in Figure 3.2. It follows that if the storage levels lie within their bounds at the start and end of loading or unloading then they must be within their bounds at all times.

In Figure 3.2 ship 1 sails to the supplier port at the beginning of the planning period, and it loads a certain amount of product at time T_1, so that the storage level of the supplier goes down because of the loading service, while the amount on board the ship goes up. Then ship 1 sails to the consumer port, and arrives at time T_2, before which time the stock level of the consumer port has gone down continuously over time because of the demand. Then it goes up because an amount of the product is unloaded from ship 1 during the service. At the same time, ship 2 arrives at the supplier after the service made by ship 1, and makes the second loading at the supplier port, after which it sails to the consumer and unloads half of the amount on board before sailing to another port. In larger examples, there are more ships and ports involved, so that the stock situation at each port is more complex than that shown in this example.

3.7 Deterministic and Stochastic Model

In realistic ocean shipping operations, much of the data needed for planning is uncertain. Because of this, deterministic models for ship routing and scheduling are sometimes inappropriate, and there is a need to develop stochastic model to deal with the uncertainties involved in the problem.

Stochastic programming is a good way to model optimization problems involving uncertainties. Whereas deterministic optimization problems are formulated with known parameters, there are always some unknown parameters in real world problems. Stochastic programming models take advantage of the probability distributions of uncertain data, and the goal is to find a policy which is feasible for the problem and optimize the expectation of some function of the decisions and the random variables.

The model introduced in the thesis deals with the situation when the rate of demand is the only uncertainty. A multi-period stochastic programming model is formulated and solved using both mixed integer programming and a decomposition approach. The case where demand varies between periods but is constant within a scenario tree node is illustrated in the right hand side of Figure 3.3 (This is the situation assumed), and the left-hand side of the

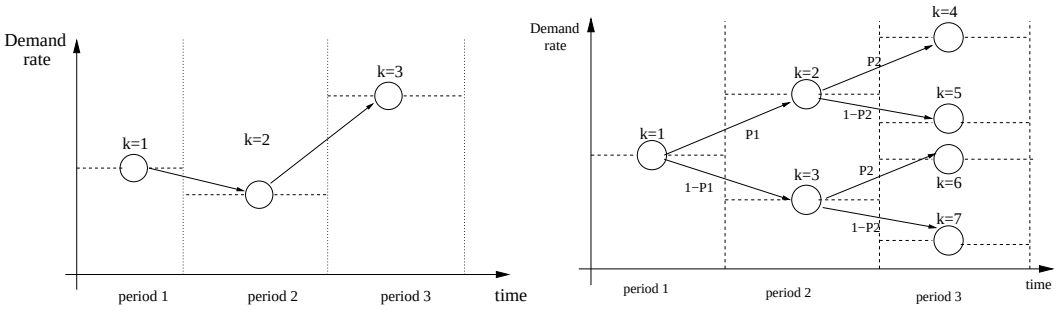


Figure 3.3: Deterministic and Stochastic

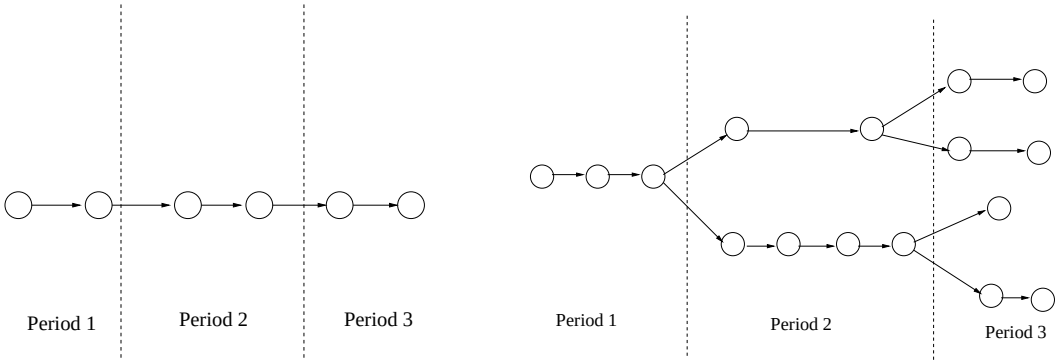


Figure 3.4: Deterministic and Stochastic

figure describes the demand situation of the deterministic model. The model manages the inventory levels at the ports and it must do this in such a way that there is no forced interruption of production (due to storage being full) and no interruption of supply to consumers (due to storage being empty).

As shown in Figure 3.3, a scenario tree is introduced for the stochastic model to represent the demand situations in each period. In a period, there are several demand situations with different probabilities. It is not known what will happen in the future until the beginning of the period, while, in the deterministic version, the demand information is known at the very beginning and a decision can be made according to this known information. Therefore, the solution of the deterministic version is a route for each ship, while the solution of the stochastic version is a tree of routes for each ship, which has the same structure as the scenario tree. Figure 3.4 provides examples of the ship routes in both the deterministic and stochastic model. In the stochastic problem, index k is used to denote the index of a node in the scenario tree and this index is used in the models throughout the thesis.

3.8 Planning period and scenario tree

In this section we introduce the planning period, how to deal with end-of-horizon of the period, scenario tree, and the relationships among them.

In our model, a whole planning period is defined and there are several small periods in the whole planning period. For example, the whole planning period can be defined as three months, and there are three periods in the planning period, one for each month. At the beginning of each small period, the demand information is revealed. In our problem, we usually define the length of a small period as a month, but we can use other length for the period. If the length of a small period is long, then a ship may visit more ports with the period. However, the ship does not need to keep sailing within the period until the end of the period. It can visit several ports and wait at some port until the next period, or it can sail to some port and arrive at the port in next period. In our decomposition model, we leave these decisions to the subproblems. We have a range of arcs in the DP network connected the ports within a period and the ports in different periods. At the end of the whole planning period, ships can be in any port. This means that ships can stop their travel anytime before the end of the horizon of the whole period and stay in the current port until the end.

In each period of the problem, there are several possible demand situations, which can be obtained from different sources, such as the historical data, the forecast from research organization and so on. Each of these situations in a period is a branch in the scenario tree of the period. For the ship routing problem, even the deterministic problem is a hard combinatorial problem, so we cannot use big sized scenario tree with a big number of branches in one period for the stochastic problem, otherwise the problem will not be solvable. But our model can be used to solve medium sized problems with reasonable scenario trees.

For the stochastic model, once new information arrives, we can rerun the stochastic model to adjust our decisions according to the new arrival information. Doing this can help us get solutions which can handle realistic situations better by using the stochastic model. However, this does not mean that the first decisions are not important since we can change our decisions later, instead we want to find better first decisions by solving the stochastic model so that we do not need to change the decisions a lot which may cause more costs. Therefore, scenario tree used for the stochastic model here should well predict the future situations. Therefore, in order to get better solutions by the stochastic model, we can consider shorter planning period with more branches in each small period and rerun the model from time to time to adjust the schedules.

3.9 Repeated Deterministic Method Based on Scenario Tree

This section introduces a heuristic method which is called the repeated deterministic method in this thesis. This method can be used to deal with uncertainties by using deterministic models, and provides a way of comparing the behavior of deterministic models and stochastic models. If the scenario tree is a good description of the uncertain situations in a model, this method gives a reasonable way to measure just how good the deterministic model is.

As stated in Kall and Wallace 1994, the method begins in the root node of the scenario tree, and solves the deterministic model. Here the mean of the uncertain parameter is used, and part of the deterministic solution which corresponds with the first period of the stochastic model is taken. This part of the solution is the solution of the repeated deterministic method at the start node of the scenario tree, and after this, the same process is repeated at each node at the second period of the scenario tree. By considering the decisions made in the first period, the deterministic model from period 2 onwards is solved, and part of solution for period 2 is used as the solution of the repeated deterministic method for each node in the second period. This continues until arrival at the leaf nodes of the tree.

This method gives a fair comparison between the stochastic model and the deterministic model. What may be done is a calculation of the difference between the optimal values of the two models. If this difference is close to zero, the deterministic model can be used to solve the problem, because the deterministic model has a smaller problem size and is easier to be solved compared with the stochastic model. However, an obvious difference would indicate that it would be better for us to use the stochastic model to solve the problem.

3.10 Port Visit

In many literatures in the area of transportation research, the port is used as the state of the model. Another concept, port visit, is used in some other literature about vehicle or ship routing problems which uses decomposition formulations. The port visit is a pair (i, m), where i is a port and m is the m-th visit to port i. For example if a ship is assigned a route P to travel, where P is $A \to B \to A \to C \to A \to B \to C$ for example, then port A is visited 3 times and ports B and C are visited twice. So route P can be represented by using a route of port visits, $\bar{P}$. $\bar{P}$ is $(A,1) \to (B,1) \to (A,2) \to (C,1) \to (A,3) \to (B,2) \to (C,2)$.

In the stochastic ship routing model, the port visit is defined by triple (Port i, Visit m, Scenario tree node k). This is used as the major part of the state of the model, and many objects in the model use the index combination

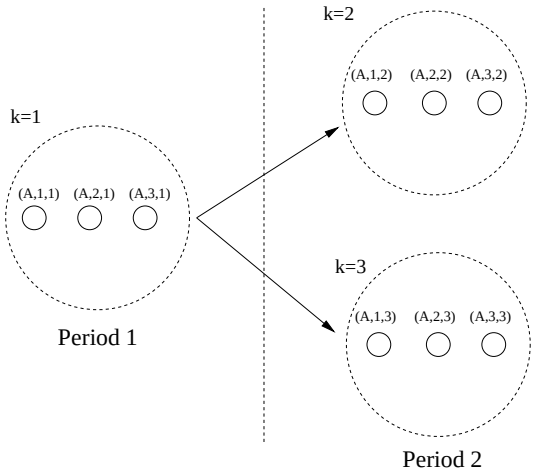


Figure 3.5: Example of port visits

(i, m, k), which means the m-th visit to port i in the scenario tree node k. An example of port visits of port A in a scenario tree is shown in Figure 3.5.

These port visits in each scenario tree node represent the possible visits to each port, but not all of these port visits need to be made by the ships. However if a port visit (i, m', k) is made by a ship, all of the port visits (i, m, k), where $m < m'$, also need to be made. Also if a port visit (i, m', k) is not made by any ship, then all of the port visits (i, m, k), where $m > m'$, cannot be made by any ship either.

In this model, if there are two port visits which refer to the same port i and in the same scenario tree node k: (i, m^1, k) and (i, m^2, k), where $m^1 \leq m^2$, the end service time for (i, m^1, k), $t^E_{im^1k}$ should be no later than the start service time for (i, m^2, k), $t^S_{im^2k}$. This corresponds to the fact that there is no overlap of two consecutive port visits to the same port in the scenario tree node and the later visit can only happen after the earlier service has finished.

Chapter 4

Mixed Integer Programming Model for Stochastic Ship Routing Problem

This chapter introduces a mixed integer programming (MIP) model for the stochastic ship routing problem. The only uncertainty involved in the model is consumer demand. The model can be solved by calling CPLEX in AMPL. Although the way of building the model is straightforward and easily to be understood, it is time-consuming to solve the model and it can only deal with discrete time periods. The model consists of two main parts, namely traveling control and storage management. The traveling control part of the model is introduced in Section 4.1 based on a deterministic situation. Section 4.2 gives the detail of the mathematical formulation of the stochastic MIP model and the computational results are shown in Section 4.3. In the section we introduce two objectives, one includes buying and selling product and the other one only deals with traveling costs. Section 4.4 presents a summary and further discussion.

4.1 Traveling Control Constraints

A key factor involved in the routing problem is finding the optimal traveling routes, and thus, the traveling control (or called generation) part is contained in almost all optimization models which deal with the routing problem. Some models use the decomposition formulations and generate routes in the subproblems, while other models use integer variables to decide the route for each vehicle.

We introduce the traveling control constraints in this section, and discuss all of the other constraints in later sections. In the MIP model, the integer variables are used to control the traveling routes of each ship. The traveling control constraints are introduced for the deterministic case here firstly,

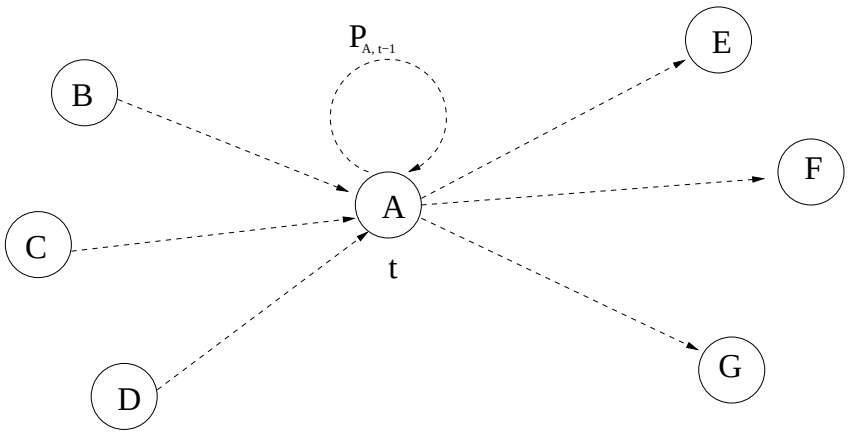


Figure 4.1: Traveling Control Balance Equation

because they may be easily understood, then the constraints are extended to the stochastic version.

In order to build the constraints, two binary variables δ_{ijtv} and p_{itv} are introduced as the ship routing control variables. δ_{ijtv}, which is used as a direction indicator in the model, is 1 if ship v, at the beginning of time period t, sails along the link from port i to port j and 0 otherwise. And p_{itv}, which is used as the ship position indicator, is 1 if ship v in time period t stays in port i and 0 otherwise. The value of these two variables indicates the traveling status and position of a ship, either sailing between two ports or staying in a port.

The basic idea of the traveling control constraints is that, given the initial condition of each ship, the traveling balance equations are used to decide the status of each ship in each period. A detailed formulation of the constraints is given as follows:

$$p_{itv} = \sum_{j \in N} \delta_{j,i,t-L_{jiv},v} + p_{i,t-1,v} - \sum_{j \in N} \delta_{ijtv} \quad i \in N,\ t \in T,\ v \in V \tag{4.1}$$

$$p_{i0v} = P_{iv}^{initial} \qquad i \in N,\ v \in V \tag{4.2}$$

$$\sum_{(i,j) \in E} \delta_{ijtv} \leq 1 \qquad t \in T,\ v \in V \tag{4.3}$$

In above constraints, L_{jiv} is the traveling time between port j and i for ship v. $P_{iv}^{initial}$ is the parameter showing the initial situation at port i for ship v. If ship v stays at port i at the beginning of period 1, $P_{iv}^{initial}$ is 1, and for any port j ($j \neq i$) $P_{jv}^{initial}$ is 0. Constraint 4.1 is the ship traveling balance equation. The value of the ship position variable in time period t, p_{itv}, equals the value of the variable at the same port i in time period $t-1$, $p_{i,t-1,v}$, plus all of the δ variables entering the port at the beginning of time period $t - L_{jiv}$

and minus all of the δ variables out of the port at the beginning of time period t. This constraint can also be represented in Figure 4.1. The constraint can also be explained in a network flow way, which is the reason the constraint is called traveling balance equation. Clearly shown in the figure, the amount of flow staying at node A at time t equals the amount of flow at the same node at time $t-1$ plus the amount of flow entering the node and minus the amount of flow out of the node in the current time period.

Constraint 4.2 initialize the position of each ship at the beginning of period 1, and constraint 4.3 ensures that only one direction can be chosen when a ship leaves a port. Furthermore, $\sum_{(i,j)\in E} \delta_{ijtv} = 0$ means that ship v will stay at the same port in time period t.

In order to understand the constraints more clearly, a small example is considered (See Figure 4.2). There are three time periods and three ports in the example, and a ship route is considered to be: $a \rightarrow b \rightarrow c$ as shown in Figure 4.2. The ship initially stays at port a in period 0, and it sets out from a to b at the beginning of period 1. It takes the ship a period to arrive at port b, and it stays at port b during period 2, then it sails from b to c at the beginning of period 3 and arrives at port c at the end of the period. The values of variable p_{itv} and δ_{ijtv} at each time period are shown in Table 4.1 and 4.2 respectively.

P_{itv}	$i = a$	$i = b$	$i = c$
$t = 0$	1	0	0
$t = 1$	0	0	0
$t = 2$	0	1	0
$t = 3$	0	0	0

Table 4.1: Example of variable p_{itv}

δ_{ijtv}	$a \rightarrow b$	$b \rightarrow c$
$t = 0$	0	0
$t = 1$	1	0
$t = 2$	0	0
$t = 3$	0	1

Table 4.2: Example of variable δ_{itv}

From the information in the tables, at time 0, ship v is at its initial position, port a, so we have $p_{a0v} = 1$ and other $p_{i0v} = 0$ $(i \neq a)$. In period 1, ship v is sailing from port a to b, so $\delta_{ab1v} = 1$. Since ship v is sailing between two ports in period 1, and it is not at any port, $p_{i1v} = 0$. In period 2, the ship stays at port b, so $p_{b2v} = 1$ with all $\delta_{ij2v} = 0$. Now the constraint 4.1 can be shown as following:

Situation	$p_{itv} = \sum_{j \in N} \delta_{j,i,t-L_{jiv},v} + p_{i,t-1,v} - \sum_{j \in N} \delta_{ijtv}$
$t = 1, i = a$	$0 = 0 + 1 - 1$
$t = 2, i = b$	$1 = 1 + 0 - 0$
$t = 3, i = b$	$0 = 0 + 1 - 1$

Table 4.3: Example of traveling balance equations

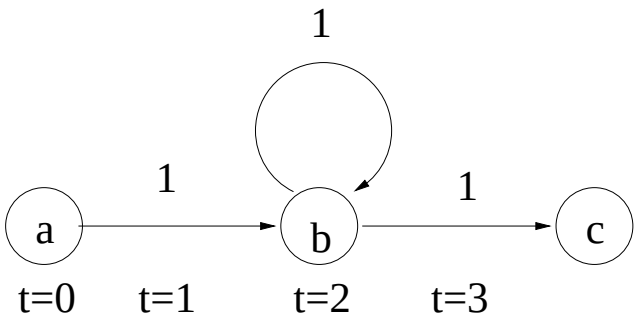


Figure 4.2: Traveling Control Equation Example

In Table 4.3, at the beginning of period 1, for example, ship v sets out from port a and sails to b. Thus $p_{a1v} = 0$ in the case that the ship spends the whole period sailing. The item of $\sum_{j \in \{a,b,c\}} \delta_{ja0v}$ is 0 because the ship does not visit any port before port a. The initial position of the ship is at port a so that we have $p_{a0v} = 1$, and $\sum_{j \in N} \delta_{aj1v} = \delta_{ab1v} = 1$ since the ship sails from a to b in period 1. Similar results can be obtained for the other situations in the above table and, in Section 4.2, these traveling control constraints are extended to the stochastic version of the model.

4.2 Mathematical Formulation of the MIP Model

This section describes the MIP formulation of the stochastic ship routing problem. First we introduce the sets and indices used in the model in Section 4.2.1. Parameters and variables are given in Section 4.2.2 and 4.2.3. The stochastic version of traveling control constraints and inventory management constraints are shown in Section 4.2.4, and Section 4.2.5 presents the objective function.

4.2.1 Sets and Indices

Let N be the set of physical ports indexed by i. There are three types of ports in the network, the consumer ports N^C, the supplier ports N^S and the dummy start port i^0. Hence there is $N = N^C \bigcup N^S \bigcup \{i^0\}$. At the beginning of

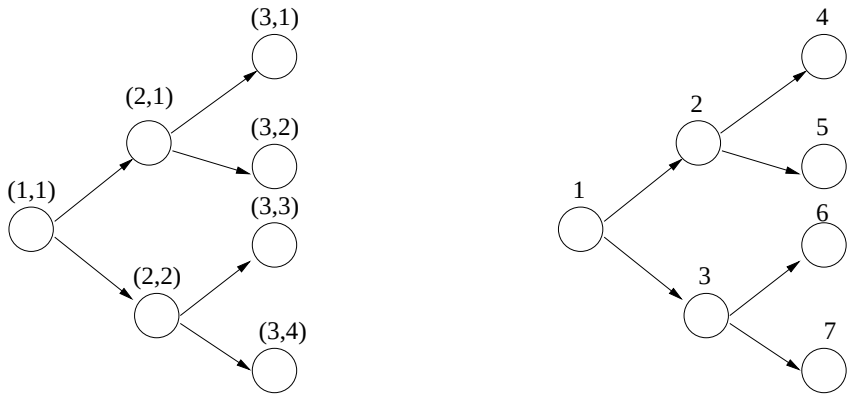


Figure 4.3: Index of Scenario Tree

the planning period, a ship can be anywhere, either a port or somewhere at sea. The information of the position of each ship is known before solving the problem. If a ship starts from a position at sea rather than a physical port $i \in N^C \bigcup N^S$, it is supposed that the dummy port i^0 is the start port of the ship. Set E represents all of the possible links between the ports in the network. $(i, j) \in E$ refers to a link from port i to port j in the network. Let V be the set of ships used in the problem indexed by v.

Since this is a stochastic problem, it is formulated based on a scenario tree. And there are several scenario tree nodes in each time period. Let T, indexed by t be the set of time period and K_t be the set of scenario tree nodes related to time period t. Each node in the tree can now be denoted by (t, a), which means the a-th node in time period t. This is shown in the first figure of Figure 4.3 as an example. Suppose that the number of possible events at node (t, n) is the same for all events at period t and is equal to $Z(t)$. Node $\bar{a}(t, n)$ is the parent node of node (t, n) in period $t - 1$. Then for node (t, n) in the tree, the corresponding parent node can be found, as well as the child node by using the following formulation:

$$\bar{a}(t, n) = \lceil \frac{n}{Z(t-1)} \rceil \tag{4.4}$$

For the sake of simplicity, the above indices are not used in the model for the scenario tree nodes. Rather K is the set of nodes in the scenario tree and $k \in K$ is used to denote the index of a node in the scenario tree, as mentioned in Section 3.7. Furthermore, parameter $a(k, t)$ is used to represent the parent node of node k t time period ago in the tree. Taking the right-hand side figure in 4.3 as an example, the values of $a(k, t)$ are represented in table 4.4.

k	1	2	3	4	5	6	7
$a(k,1)$	–	1	1	2	2	3	3
$a(k,2)$	–	–	–	1	1	1	1

Table 4.4: Example of $a(k,t)$

4.2.2 Parameters

Parameter $P_{iv}^{initial}$ indicates the initial position of ship v. If ship v stays at port i at the beginning of planning period, $P_{iv}^{initial} = 1$ and $P_{jv}^{initial} = 0$ for all $j \in N$ and $j \neq i$. If ship v starts from a position in the sea, $P_{i^0v}^{initial} = 1$ and $P_{iv}^{initial} = 0\ \forall i \in N^C \bigcup N^S$. The traveling cost for any ship along a link (i,j) is given by the parameter C_{ij}^{travel}. The parameter L_{ijv}^{ship} represents the traveling time for ship v sailing along arc (i,j). The capacity of each ship is given by G_v^{ship}.

When a ship arrives at a supplier port, it loads an amount of the product from the storage of the port, and when it arrives at a consumer port, an amount of the product is discharged from the ship. The loading limit in port i is the parameter E_i^{load}. There is a storage space at each port with the upper limit $\bar{S}_i$ and lower limit $\underline{S}_i$.

In the supplier ports, an amount of the product is injected into the storage as a supply in each time period, and the parameter R_i^{supply} is taken as the fixed rate of injection at supplier i. In the consumer port, an amount of the product is taken out of storage as demand. D_{ik} is used to denote the demand at consumer port i at scenario tree node k. Once the demand for a consumer port cannot be satisfied, an amount of gas would be bought from other resource at a higher price. Here the parameter $C^{penalty}$ gives the unit purchase price from other resources when there is no sufficient supply. Since the price from other resources would be much higher, this cost can be regarded as a penalty cost for failing to give enough supply. For each port storage, there is an initial stock level, S_i^0, at the beginning of the planning period.

As discussed in section 4.2.1, parameter $a(k,t)$ is taken to represent the parent node, t time periods ago, of node k in the tree. This parameter gives the structure of the scenario tree. Parameter P_k denotes the cumulative probability associated with scenario node k.

4.2.3 Variables

Three types of variables are used in the formulation, namely, the ship routing control variables, the on board monitor variables and the storage monitor variables.

The binary variable δ_{ijvk} is 1 if ship v at the beginning of the period which includes scenario tree node k sails along the link from port i to port

j and 0 otherwise. The binary variable p_{ikv}, the ship position variable, is 1 if ship v in scenario tree node k stays at port i and 0 otherwise. These two binary variables, named ship routing control variables, indicate the status and position of a ship in each scenario tree node. Note that in the stochastic problem, the index of time period t is substituted by scenario tree node k.

The amount of product loaded or discharged at port i from ship v at the scenario tree node k is given by the variable q_{ivk}. The amount of the product on board ship v at node k is monitored by the variable g_{vk}. These two variables give the information of the changes of the amount on board and the current amount on board. This can ensure that the amount on board does not exceed the ship's capacity and that the loading or unloading amount does not exceed the availability on board by using the variables, so called the on board monitor variables.

The third type of variables is called storage monitor variables. The variable h_{ik} represents the amount of the product in storage at port i in node k. The amount of product taken out of storage as demand at consumer node i in scenario tree node k is denoted by z_{ik}. This variable does not need to be equal to the demand at each scenario tree node, because the product can be obtained from other resources at a higher price. Moreover, the variable b_{ik} is the amount of gas bought from other resources at port i in scenario tree node k when there is an insufficient supply. This variable can keep the problem feasible in each demand scenario node of the scenario tree.

4.2.4 Constraints

The traveling control constraints for the deterministic situation is discussed in Section 4.1. Here the traveling control constraints are firstly presented for the stochastic model as constraints 4.5 – 4.7.

$$p_{ikv} = \sum_{j \in N} \delta_{j,i,a(k,L^{ship}_{jiv}),v} + p_{ia(k,1)v} - \sum_{j \in N} \delta_{ijkv}, \ i \in N, k \in K, v \in V \quad (4.5)$$

$$p_{i0v} = P^{initial}_{iv}, \qquad i \in N, v \in V \quad (4.6)$$

$$\sum_{(i,j) \in E} \delta_{ijkv} \leq 1, \qquad k \in K, v \in V \quad (4.7)$$

In constraints 4.5 and 4.7, index k is taken as the scenario tree node, instead of index t, the time period, in constraints 4.1 and 4.3. According to the tree structure of the problem, the previous nodes of node k in the same scenario can be found by parameter $a(k, T)$. Assuming that ship v sails from port j to port i and arrives at port i in scenario tree node k. Because L^{ship}_{jiv} is the traveling time along link (j, i) by ship v, ship v has to set out from port j and sail to port i L^{ship}_{jiv} time periods ago so that it can arrive at port i at scenario node k,

and the corresponding parent node of the scenario node k, L_{jiv}^{ship} periods ago, is node $a(k, L_{jiv}^{ship})$. Therefore, if ship v sets out from j and sails to i in node $a(k, L_{jiv}^{ship})$, variable $\delta_{j,i,v,a(k,L_{jiv})}$ is 1 and $\delta_{j,i,v,a(k,t)}$ $(\forall t = \{1, 2, ..., L_{jiv} - 1\})$ is 0. Constraints 4.6 and 4.7 are almost the same as the constraints in the deterministic situation except for the index k in constraint 4.7.

Now the storage management constraints are introduced. There are two main things of interest in terms of storage management, namely the monitoring of the stock level of the storage at each port and the monitoring of the amount of product left on board the ships over time periods. The storage balance equation and initial storage value at consumer ports are given below:

$$h_{ik} = h_{i,a(k,1)} + \sum_{v \in V} q_{ivk} - z_{ik}, \quad i \in N^C,\ k \in K \tag{4.8}$$

$$h_{i,0} = S_i^0, \quad i \in N^C \tag{4.9}$$

Constraint 4.8 describes the stock situation at each consumer port. The stock level at port i in node k is the stock level at the same port in the parent node of k in the previous period, $a(k, 1)$, plus the amount of commodity unloaded from each ship, $\sum_v q_{ivk}$, and minus the amount of demand taken out of storage in node k, z_{ik}. Constraint 4.9 gives the initial storage level at each consumer port, and similar storage balance equations and initial condition constraints for supplier ports are given in Constraint 4.10 and 4.11:

$$h_{ik} = h_{i,a(k,1)} + R_i^{supply} - \sum_{v \in V} q_{ivk}, \quad i \in N^S,\ k \in K \tag{4.10}$$

$$h_{i,0} = S_i^0, \quad i \in N^S \tag{4.11}$$

Constraint 4.10 is slightly different from constraint 4.8. The stock level at supply port i in node k is the stock level at the same port in the parent node k of the previous period, plus the amount of product injected into the storage in current node, and minus the amount loaded onto the ships. Clearly, the stock levels at any time for both consumer and supplier ports should not exceed the maximum capacity of the storage or be below the lower bound of the storage. This can be shown in the capacity constraints of storage:

$$\underline{S}_i \leq h_{ik} \leq \bar{S}_i \quad i \in N,\ k \in K \tag{4.12}$$

Then the amount of commodity taken out of storage at port i in node k, z_{ik}, can be calculated by the demand constraint 4.13:

$$z_{ik} + b_{ik} = D_{ik}, \quad i \in N^C,\ k \in K \tag{4.13}$$

Constraint 4.13 indicates that the consumer demand in the model can be satisfied by two resources: from storage and from another resource at a higher cost. Since it is possible, in some situations, that the consumer demand cannot be satisfied by a ship transport operation, the stock level will then break the lower bound of the storage. In this case, the product can be bought from another resource so as to meet the demand requirement and keep the stock level beyond the lower bound of the storage. However, a higher cost will be incurred by doing this, since there will be a penalty price for insufficient supply in the objective function.

Next, constraints to monitor the product level on board the ships are introduced. The only way each ship can get products is loading them from the supplier ports. The amount loaded by ship v from the supplier ports in node k can easily be calculated by $\sum_{i \in N^S} q_{ivk}$, and the amount on board ship v is decreased when the ship arrives at a consumer port and discharges an amount of the product in the port storage. The amount discharged at consumer ports in node k can be calculated by $\sum_{i \in N^C} q_{ivk}$. Therefore, the balance equation of amount on board each ship is shown in 4.14

$$g_{vk} = g_{v,a(k,1)} + \sum_{i \in N^S} q_{ivk} - \sum_{i \in N^C} q_{ivk} \quad v \in V,\ k \in K \tag{4.14}$$

The other two constraints 4.15 and 4.16 of the model are given as follows:

$$g_{vk} \leq G_v^{ship} \quad v \in V,\ k \in K \tag{4.15}$$

$$q_{ivk} \leq G_v^{ship} p_{ivk} \quad v \in V,\ k \in K,\ i \in N \tag{4.16}$$

Constraint 4.15 shows that, at any time, the amount on board a ship cannot exceed the capacity of the ship, and constraint 4.16 indicates that a ship can only be loaded or unloaded when the ship arrives at a port, and the loading or unloading quantity cannot exceed the capacity of the ship.

4.2.5 Objective Function

The objective function of the MIP model used here is to minimize the total traveling costs, shown as below.

$$\min \sum_{k \in K} P_k \Big(\sum_{v \in V} \sum_{(i,j) \in E} C_{ij}^{travel} \delta_{ijvk} + \sum_{i \in N^C} C^{penalty} b_{ik} \Big) \tag{4.17}$$

Then constraints 4.5 – 4.16 and objective 4.17 together formulate the MIP stochastic ship routing model, and by substituting the index k to index t, the stochastic model can be transferred to the deterministic model.

4.3 Computational Results

In this chapter, the ship routing problem is formulated as an MIP model. In order to test both deterministic and stochastic models built here, a set of test examples are generated. The models were written in AMPL and solved directly by calling the commercial optimization software package CPLEX 10.0. The geographical positions of the ports were randomly generated and the distances between the ports were calculated a priori. The traveling costs and time are roughly in proportion to the distances, but only the integer value is taken for these parameters. Since there are difficulties involved in solving the integer programming problem, the problem can only be solved for smaller sized examples in this thesis.

Ex.	Ports	Ships	Periods	CPU time	B&B Nodes	MIP iters
mipd1	5	3	11	1.3	932	12256
mipd2	5	4	11	25.7	26022	309591
mipd3	5	3	12	52.9	65016	606094
mipd4	5	4	12	6167.4	4372425	41778444
mipd5	6	4	11	1476.5	746557	7744434
mipd6	6	4	12	25278.5	18385655	163104917

Table 4.5: Deterministic Example Results (MIP)

Table 4.5 shows characteristics of the problem and the solutions for the deterministic models. The 'Ports' column shows the number of ports considered in each example, some of which are consumer ports, and others are supply ports. The 'Ships' column shows the number of ships involved in the examples, and the number of periods is shown in the 'Periods' column. Since this MIP model can only handle the discrete time, these periods are the smallest time unit in the model. For example, the traveling time between two ports is two periods, and a ship stays in a port for a period, and so on. The 'CPU time' column gives the solving seconds to obtain the optimal solution, and the 'B&B Nodes' and 'MIP iters' columns show the number of branch-and-bound nodes in the branch-and-bound tree for searching the integer solution and the number of MIP simplex iterations used respectively.

Table 4.6 shows the examples used for the stochastic models. The numbers given in the 'Periods' column are similar to those given in table 4.5. The 'Scen. nodes' column shows the number of scenario tree nodes in each example, and within each scenario tree node, small time slots are again used in the model

Ex.	Ports	Ships	Scen. nodes	Periods	CPU time	B&B Nodes	MIP iters
mips1	5	2	3	16	8	4184	115981
mips2	5	2	3	16	1251.2	351682	16517968
mips3[a]	5	2	3	16	248.7	125114	4451257
mips4	5	2	7	24	150000[b]	4291924	379366387
mips5	5	2	7	24	108086	4876345	353237817
mips6[a]	5	2	7	24	86708.9	3923915	2799088058
mips7	6	2	7	24	138260	3421307	387788785
mips8	6	2	7	24	95943.7	4344379	283635043
mips9[a]	6	2	7	24	122798	3217502	301691099

[a] Ex mips1-3, mips4-6 and mips7-9 have the same network but different initial storage and demand rates
[b] Solving time reaches the upper limit, problem is not solved to optimal

Table 4.6: Stochastic Example Results (MIP)

so that a new scenario tree needs to be built for solving the stochastic model. Both the number of scenario tree nodes and the number of periods are used to build the structure of the scenario tree. An example of the scenario tree for example *mips*5 is illustrated in Figure 4.4.

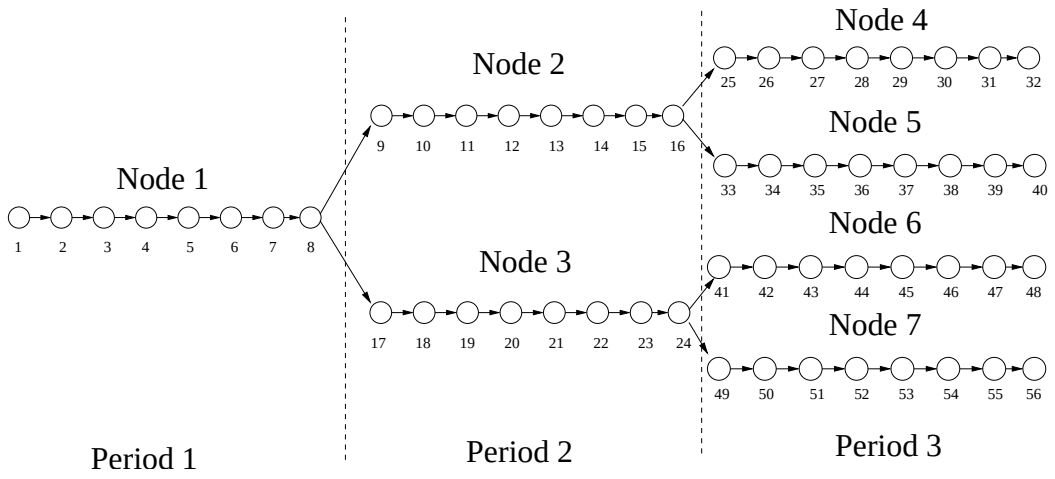


Figure 4.4: Index of Scenario Tree

The figure shows seven scenario tree nodes in the tree, which indicate different demand situations. Within each scenario tree node, the time period is divided into 8 small time slots, each of which is associated with a node in the new scenario tree so that there are 8 nodes within a big scenario tree node. Therefore, there are 7 big parts in the scenario tree and each part contains 8 nodes. In the example, there are 56 nodes in the tree in total. Based on the scenario tree of the figure, the parameter $a(k, T)$ can be specified, which describes the structure of the scenario tree for the example.

In table 4.6, examples mips1–mips3, mips4–mips6 and mips7–mips9 have

the same physical ports layout and scenario tree structure, but different initial inventory levels and demand rates at each port. It can be seen from the results in both tables that it is very time-consuming to solve either deterministic or stochastic ship routing problems by using the MIP formulation. For instance, it takes 138260 seconds (about 38 hours) to solve example *mips*7 to optimal. Example *mips*4 reaches the upper limit of the solving time, since it is not solved to optimal. According to computational experience, the solving time increases sharply when the number of ports, number of ships, and size of the scenario tree is increased. Even if only the initial storage levels or demand rates for the same problem were changed, the problem would become harder and take much more time to be solved.

4.4 Summary and Discussions

This chapter introduced an MIP formulation for the stochastic ship routing problem. The model is divided into two parts, traveling control part and inventory monitoring. Two integer variables, p_{ikv} and δ_{ijkv} are used in the traveling control constraints to decide the ship's position and status. A set of continuous variables is used in the inventory monitoring constraints to record the stock level at each port, the transfer amount either from supplier to ship or from ship to consumer, and the amount of product on board each ship.

In order to test both deterministic and stochastic models, a group of examples were created, and these differed from one another in terms of the number of ports, the number of ships, scenario tree structures, initial stock levels and demand rate situations. All of these examples were written in AMPL and solved by calling CPLEX 10.0. CPU seconds are reported, as well as the number of branch-and-bound nodes and the number of MIP simplex iterations for solving these examples. The ship routing problem with inventory management problem is a difficult combinatorial problem, and it is quite time-consuming to solve the problem using the MIP formulation, although it is very straightforward and easy to understand if the problem itself is built in the way of MIP formulation. Thus the decomposition formulation will be built and a solving algorithm for the stochastic ship routing problem will be presented in Chapter 5 in order to solve a relatively large size problem within a reasonable time.

Chapter 5

Decomposition Formulation for Stochastic Ship Routing Problem

5.1 Introduction

This chapter introduces a decomposition formulation of the stochastic ship routing problem, which is solved by column generation method, and the optimal integer solution is found by branch-and-bound algorithm. Compared with the MIP model introduced in Chapter 4, which is straightforward but time consuming to solve, the model in this chapter is divided into a master problem and several subproblems, and can be used to solve relatively big size examples within reasonable time.

For the stochastic ship routing problem, a scenario tree is introduced to represent the demand situation in each time period. The solution of the problem is a tree of routes with start service time and quantity loaded information at each visited port for each ship. In the rest of the chapter, index k is used to indicate the scenario tree node. Each port i in a scenario tree node k is divided into several visits, for example, (i, m, k) is the m-th visit to port i in the scenario tree node k. So port visit (i, m, k) is used as the state of the model instead of physical port i. Many variables and parameters in the model use the index (i, m, k). The integer solution of the model is searched along a branch-and-bound tree, and in each branch-and-bound tree node, a relaxed LP problem is solved by solving a restricted master problem and a set of subproblems iteratively, and this is called column generation method. The master problem is solved to find the best ship schedules based on the current set of generated schedules and provide dual variable information to update the subproblems. The subproblems, one for each ship, are formulated as Stochastic Dynamic Programming (SDP) problem with time windows to generate the most promising tree of routes for the master problem.

In later sections, some assumptions are made for the model in Section 5.2 and the overall solution framework is given, including branch-and-bound algorithm and column generation process in Section 5.3. Details of master problem is shown in Section 5.4 and reduced cost formulation is given in Section 5.5. Section 5.6 and 5.7 present the DP formulation of the subproblem and the corresponding DP network. Set labeling method for solving the subproblem is introduced in Section 5.9. A discussion of cycle elimination is in Section 5.10. Then the branch-and-bound strategy is presented in Section 5.11 and the computational results in Section 5.12. A brief conclusion and remarks are given in Section 5.13.

5.2 Assumptions

The ocean transportation problem is so complex that it is impossible 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. Some of these assumptions are used here to simplify the problem, but they are not the restrictions to our model. Our model is able to deal with the situations without these assumptions. But other assumptions may be necessary to build the model.

- **At each consumer port, the rate of demand is constant within a scenario tree node, but can change between nodes, although the demand of each consumer port is uncertain.** In this model, there are several possible demand situations in each period, and this is represented in the scenario tree as the branches in each period with corresponding probabilities. In each demand branch, the demand rate is a constant throughout the period, but would be different between different branches.

- **At each port, loading and unloading rates are constant.** In realistic operations, the loading or unloading rate may be varied for different ships at the same port. However, we here assume that it is fixed for a port throughout the whole planning period to simplify the problem but these rates can be different in different ports.

- **At most one ship can be loading or unloading in a port at any given time.** This assumption avoids the overlap of services at a port. When a ship arrives at a port and another ship is loading or unloading at the same port, the ship has to wait outside the port until the current service is completed. We assume that there is no waiting cost for each ship at any time.

- **For each ship the travel time and cost between any two ports are fixed.** This is the assumption only for this chapter, and a variable

speed model is introduced in Chapter 6. We also assume that ships sail in a straight line between any two ports in the model, although in the realistic situation ships may be not able to sail in a straight line.

- **A service at a port must start and finish within a period.** However, this is not a limitation, since the service can continue without a break in the following period.

5.3 Solution Framework

A branch-and-price algorithm is used to search for the optimal integer solutions. In each branch-and-bound node, the node problem is decomposed to a master problem and several subproblems, one for each ship. Each column in the master problem corresponds to a tree of schedules for a ship. And the node problem is solved by a column generation method which requires only a finite number of columns.

At any stage in the solution of a master problem, a (finite) subset of the columns will have been generated. This problem, called a restricted master problem in Ahuja et al. 1993, is solved and the shadow prices of the constraints are then used to determine the most negative reduced cost from among the un-generated 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 routes for that ship. If this added as a column to the master problem, it would have the smallest a negative reduced cost. This procedure continues until no column with negative reduced cost can be generated, at which stage the master problem for the Branch and Bound node has been solved.

If found optimal solution at a branch-and-bound node is not integer, the branching variable is chosen, and child nodes of the current branch-and-bound node are generated. The search is stopped when there is no node in the branch-and-bound tree, and the stored integer solution is the final solution.

5.4 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 as mentioned before. For any ship, there are a set of trees of schedules for it. The problem is to choose a 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} – loaded/unloaded quantity by ship v in (i, m, k) if the 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 – loading/unloading rate at port i
M – the maximum number of visits to any port in a scenario tree node
D_{ik} – demand rate in port i in node k
S_i^0 – 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
I_i – 1 if i is customer port, -1 if i is supplier port

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 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.

Figure 5.1 shows an example of a tree of routes for 2 ships. The red tree is made by ship 1, while ship 2 makes the blue one. For instance, ship 1 makes

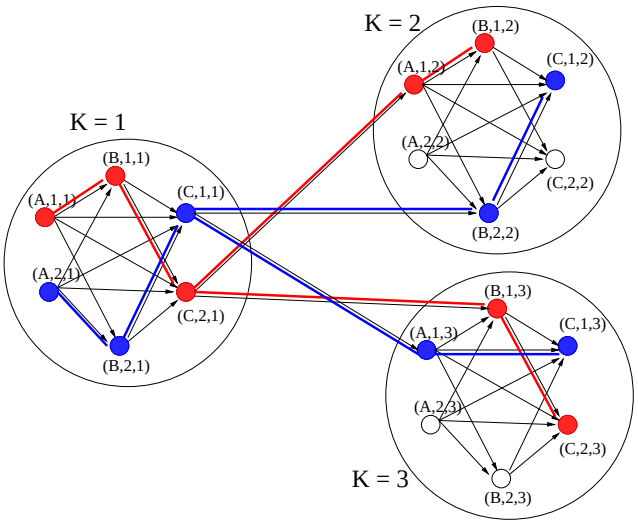
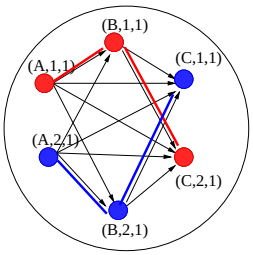


Figure 5.1: Example of tree of routes

K = 1



	A_{rsimk}		Q_{rsimk}		T_{rsimk}	
	s1	s2	s1	s2	s1	s2
(A,1,1)	1	0	800	0	1.0	0.0
(A,2,1)	0	1	0	800	0.0	2.0
(B,1,1)	1	0	800	0	3.5	0.0
(B,2,1)	0	1	0	800	0.0	4.0
(C,1,1)	0	1	0	800	0.0	6.0
(C,2,1)	1	0	800	0	6.5	0.0

Figure 5.2: Parameter example

the first visit to port A followed by a visit to port B and a visit to port C in scenario node 1. Then it visits port A and B in scenario node 2, while it visits port B and C in node 3. Figure 5.2 shows the detail values of parameters A_{svimk}, Q_{svimk} and T_{svimk} for both ships in the scenario node 1. For example, ship 1 makes port visit $(A, 1, 1)$, $(B, 1, 1)$ and $(C, 2, 1)$ in scenario node 1, so the corresponding values of A_{svimk} for these port visits are 1 and all others are 0.

The parameter M is the maximum number of visits to a port in a time period, but a port does not need to be visited exactly M times in each scenario node. In a scenario tree node, a port can be visited any number of times up to M. We consider inventory at both supply and consumer port, so parameter I_i here is used to separate these two different types of ports so that we can use a single constraint for both types in the master problem.

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 loaded/unloaded in port visit (i, m, k)
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}

Variable y_{imk} is a binary variables here. But when we solve the relaxed master problem in a branch-and-bound node, the integrality requirement will be relaxed. If variable x_{sv} is 1, ship v chooses to take schedule tree s. However, x_{sv} do not need to be integer value in the model. Two x_{sv} variables, x_{s_1v} and x_{s_2v}, may represent the same physical tree of routes but have different loading quantities and start service times at each port visit. If $x_{s_1v} + x_{s_2v} = 1$, then the solution is still regarded as integer feasible. Ship v chooses to take the physical tree of routes, and uses the average loading quantities and start service times of schedule s_1 and s_2 as the loading quantity and start service time at each port visit on the tree of routes. Variable q_{imk}, t^S_{imk} and t^E_{imk} provide the information for each port visit. h^S_{imk} and h^E_{imk} are used to record the stock level at the beginning and end of a service. These two variables should be bounded with the upper and lower limit of the corresponding storage.

Formulation of Master Problem

$$\min \quad \sum_{v\in V}\sum_{s\in R_v} C_{sv}x_{sv} \tag{5.1}$$

$$\sum_{v\in V}\sum_{s\in R_v} A_{svimk}x_{sv} = y_{imk} \quad \forall (i,m,k)\in P \tag{5.2}$$

$$\sum_{v\in V}\sum_{s\in R_v} Q_{svimk}x_{sv} = q_{imk} \quad \forall (i,m,k)\in P \tag{5.3}$$

$$\sum_{v\in V}\sum_{s\in R_v} T_{svimk}x_{sv} + (1-y_{imk})B_k = t^S_{imk} \quad \forall (i,m,k)\in P \tag{5.4}$$

$$\sum_{s\in R_v} x_{sv} = 1 \quad \forall v\in V \tag{5.5}$$

$$t^E_{imk} = t^S_{imk} + q_{imk}/W_i \quad \forall (i,m,k)\in P \tag{5.6}$$

$$t^E_{i,m-1,k} \le t^S_{imk} \quad \forall (i,m,k)\in P,\ m>1 \tag{5.7}$$

$$y_{imk} \ge y_{i,m+1,k} \quad \forall (i,m,k)\in P \tag{5.8}$$

$$h^E_{imk} = h^S_{imk} - (t^E_{imk} - t^S_{imk})I_iD_{ik} + I_iq_{imk} \quad \forall (i,m,k)\in P \tag{5.9}$$

$$h^S_{imk} = S^0_i - t^S_{imk}I_iD_{ik} \quad \forall i\in N,\ m=1,\ k=1 \tag{5.10}$$

$$\begin{aligned} h^S_{imk} = {} & h^E_{i,M,a(k)} - (B_{a(k)} - t^E_{i,M,a(k)})I_iD_{i,a(k)} \\ & - (t^S_{imk} - B_{a(k)})I_iD_{ik} \quad \forall i\in N, m=1, k>1 \end{aligned} \tag{5.11}$$

$$h^E_{iMk} - (B_k - t^E_{iMk})D_{ik} \ge 0 \quad i\in N,\ k\in K^T \tag{5.12}$$

$$h^S_{imk} = h^E_{i,m-1,k} - (t^S_{imk} - t^E_{i,m-1,k})I_iD_{ik} \quad \forall (i,m,k)\in P\ m>1 \tag{5.13}$$

$$\underline{S}_i \le h^S_{imk},\ h^E_{imk} \le \bar{S}_i \quad \forall (i,m,k)\in P \tag{5.14}$$

$$\{x_{sv} : x_{sv} > 0, \forall s\} share\ same\ physical\ tree\ of\ routes,\ \forall v \tag{5.15}$$

$$y_{imk}\in\{0,1\} \quad \forall\ (i,m,k)\in P \tag{5.16}$$

In (5.1) we minimize the total expected costs. Constraints (5.5) and the integrality of the x_{sv} ensure that exactly one tree of schedules is selected for each ship. Constraints (5.2) 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 (5.3) calculates the loading or unloading quantity and constraint (5.4) calculates the start of service time for each port visit. If port visit (i,m,k) occurs, then the first term in (5.4) 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 (5.6) calculates the end of service time and constraint (5.7) guarantee 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 (5.8) 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

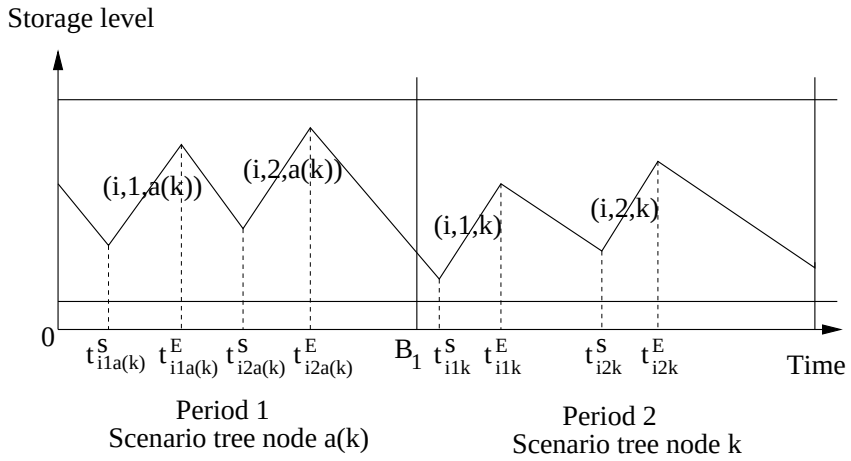


Figure 5.3: Storage level Ex

(5.9)-(5.14) 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.

An example of storage level for a consumer port i is shown in Figure 5.3. We record the storage levels at start service time and end service time for each service and these levels are bounded between the upper and lower bound of the storage tank by storage constraints. For the planning period we need to consider different time intervals to calculate the storage levels when we build inventory constraints here. These situations are listed below:

- **From the very beginning of the period 1 to the first visit in the period** In this situation, the stock level at the start service time t^S_{i11} equals the initial storage level S^0_i at port i minus the demand taken out of the storage during time interval $[0, t^S_{i11}]$. Constraint 5.10 represents this situation.

- **During a service** Constraint 5.9 shows the storage situation during a service. At a consumer port, for instance, storage level at the end of a service m equals the level at the start of the service plus the quantity discharged from ship and minus the demand taken out of the storage during the service. In constraint 5.9, parameter I_i is 1 or -1 as a sign to tell consumer or supplier port, which allows us to use the same constraint for both types of ports in the model. In the example, this shows the situation between start service time and end service time.

- **Between two consecutive services within a period** In this situation, the stock level at the start service time for the second service

equals the stock level at the end service time for the first service minus the demand taken out of the storage during the time interval, and this is presented in Constraint 5.13.

- **Between the last service in a period and the first service in next period** Constraint 5.11 presents the situation between the last service in the scenario tree node $a(k)$ and the first service in the scenario tree node k, where $a(k)$ is predecessor node of k. This shows the situation of stock across the boundary of a period in the example.

- **After the last service of overall planning period** Constraint 5.12 calculates the storage level at the end of the last service in the last period.

If only m ($m < M$) port visits occur for a port within a period, the loading quantities at port visit $m + 1..M$ are zero. We call such port visits 'dummy port visits'. According to the Constraint 5.4, the start service times for these port visits are on the boundary of the period. Also the stock levels related to these dummy visits are the same as the stock level at the end of the period, which equals the stock level at the end service time of the m-th visit minus the demand taken out of the storage from the end service time to the boundary of the period. There is a special case in our model, when there is no visit to a port in a period. In this case, all M visits are dummy visits within that period and happens on the end boundary of the period. The corresponding stock levels on these dummy port visits give the storage situation at the end of the period.

From the formulation of the master problem we can see that the constraints 5.2–5.5 are similar to the constraints in set partitioning models. However there are in addition a group of inventory constraints in the master problem. The form of the matrix of the master problem is shown in Figure 5.4.

The blue block on the left upper corner is for the constraint 5.2 – 5.4. The pink block is for all the inventory constraints. The purple block is the initial columns with huge objective costs. These columns are used to give an initial feasible solution for the column generation method and they will never be chosen in the optimal solution. The columns generated from the ship subproblems are added in the yellow area of the matrix. The rows in the bottom of the yellow area are the constraint 5.5, which indicates the ship related to each added column.

5.5 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

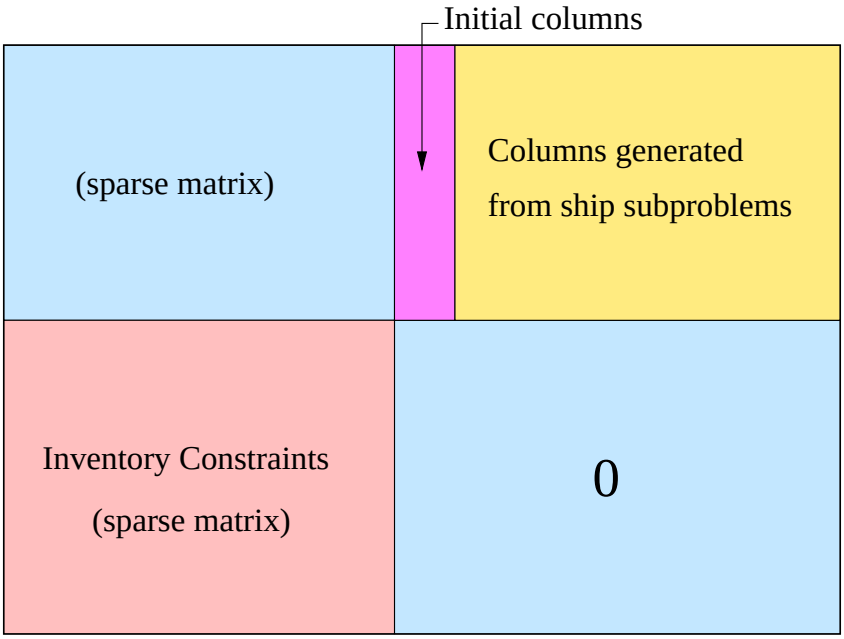


Figure 5.4: Matrix of master problem

(5.11)–(5.5) respectively. The reduced cost $\bar{C}_{sv}$ can then be calculated as following:

$$\hat{C}_{sv} = C_{sv} - \sum_{i,m,k} (A_{svimk} d^A_{imk} + Q_{svimk} d^Q_{imk} + T_{svimk} d^T_{imk}) - d^S_v \tag{5.17}$$

$$= \sum_{(i,m,k)-(i',m',k') \in E_s} P_{k'} C_{ii'v} - \sum_{(i,m,k) \in N_s} (d^A_{imk} - d^Q_{imk} Q_{svimk} + d^T_{imk} T_{svimk}) - d^S_v \tag{5.18}$$

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.

Here, we represent the reduced cost for each arc of the network in formula 5.18, then we can formulate the subproblem objective and update the edge costs in the networks of subproblems according to formula 5.18.

5.6 Ship Subproblems

The parameters Q_{svimk} and T_{svimk} as well as set E_s and N_s in (5.18) represent the route information generated by subproblems and is not given in advance. So we can introduce two variables q_{svimk} and t^S_{svimk} in the subproblems for

loading quantity and start service time respectively. In subproblems we try to find the values for these variables and the set of E_s and N_s so as to minimize the reduced cost. We denote r to be the physical tree of routes of schedule s. Then the details about the tree of routes r can be described by E_s and N_s. For a ship v, the objective of the subproblem can be formulated as following:

$$\begin{aligned}\bar{C}_v = \quad & \min_r \min_q \min_{t^S} [\sum_{(i,m,k)\rightarrow(i',m',k')\in E_s} P_{k'} C_{ii'v} \\ & - \sum_{(i,m,k)\in N_s} (d^A_{imk} + d^Q_{imk} q_{svimk} + d^T_{imk} t^S_{svimk})] - d^S_v \end{aligned} \tag{5.19}$$

In formulation 5.19, 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 presented in formula 5.18. d^S_v in 5.18 does not need to be considered in the subproblems. It can be subtracted from the objectives after solving the subproblems.

The ship subproblems 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, which 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.

The DP network is described in Section 5.7 and the DP formulation is given in Section 5.8.

5.7 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

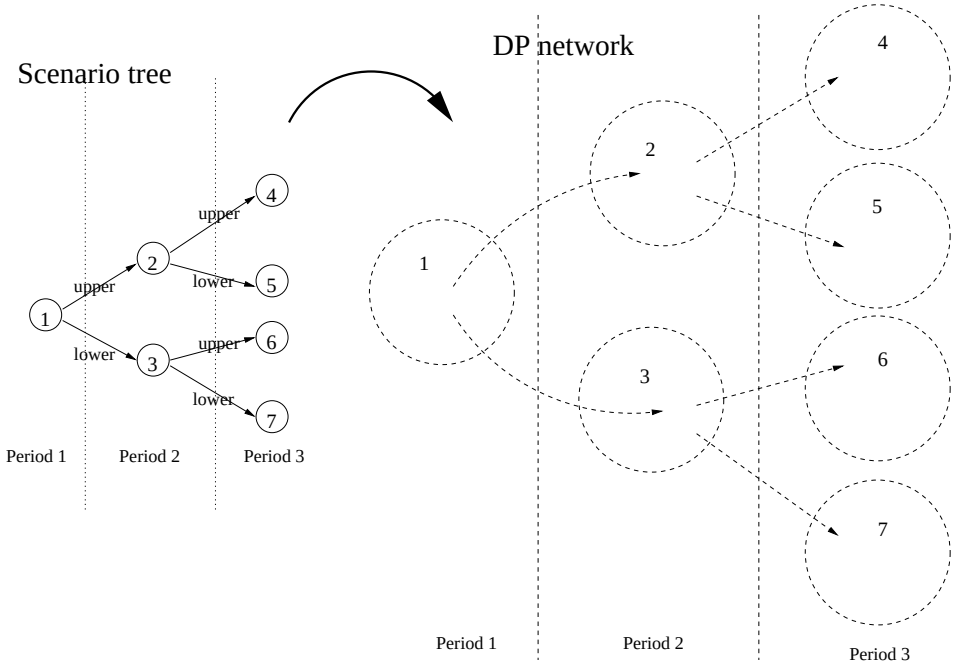


Figure 5.5: Structure of DP network

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 5.5 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 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 5.6 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. A dummy final node is also necessary in the DP network for solving the problem

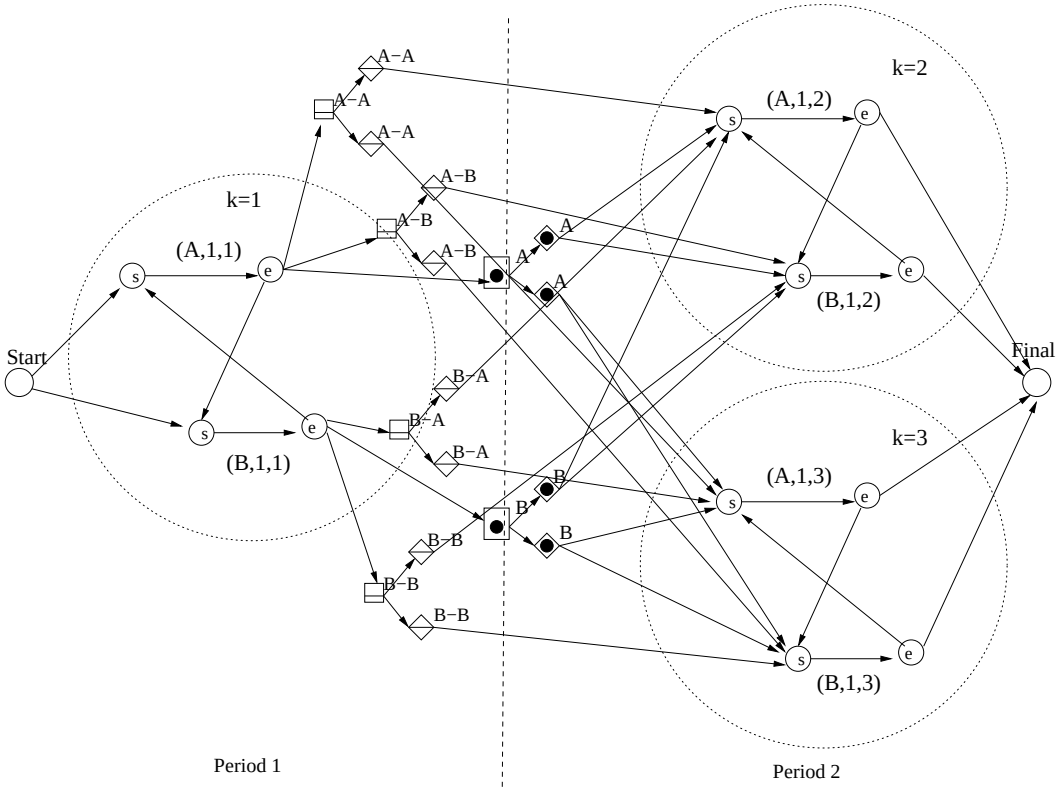


Figure 5.6: DP network with maximum of one visit to each port in each demand scenario part

along the network. We can put an increasing cost function of time on the final dummy node so as to encourage ships to finish service as early as possible. The different types of nodes in the network are listed in the table below:

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 $\blacklozenge$ nodes, one for each demand scenario part. The decisions made on these $\blacklozenge$ 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 a 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 5.6, 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 5.7. When we update the cost function of each node, different situations of different time points and quantities need to be considered.

A more detailed DP network in a scenario part is given in Figure 5.8. There are two ports in this example, each of them has two visits within the period. By using this example, we can see the details about the nodes and arcs involved in the DP network. As discussed before, there are two types of nodes related to a port visit, a start service node Ⓢ and an end service node Ⓔ . In the example, an arc from start service node Ⓢ to end service node Ⓔ represents a loading or unloading service. 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 5.9 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

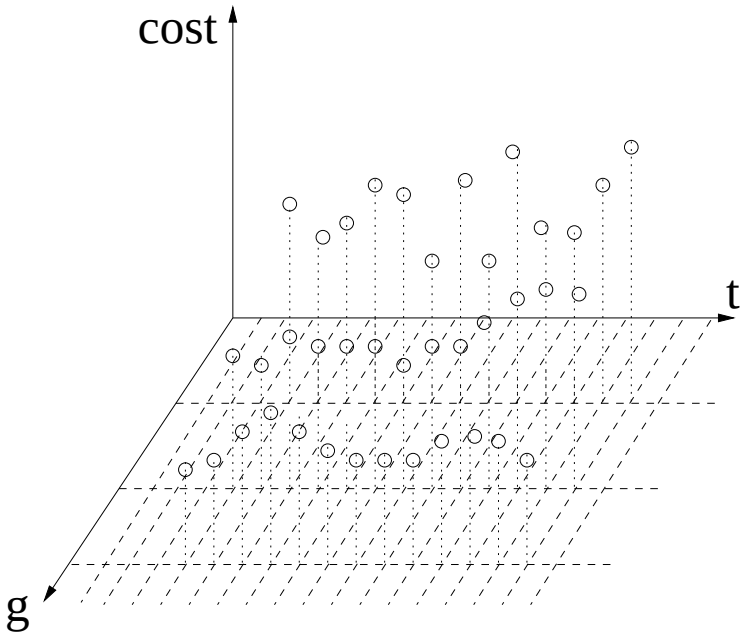


Figure 5.7: General cost function on node

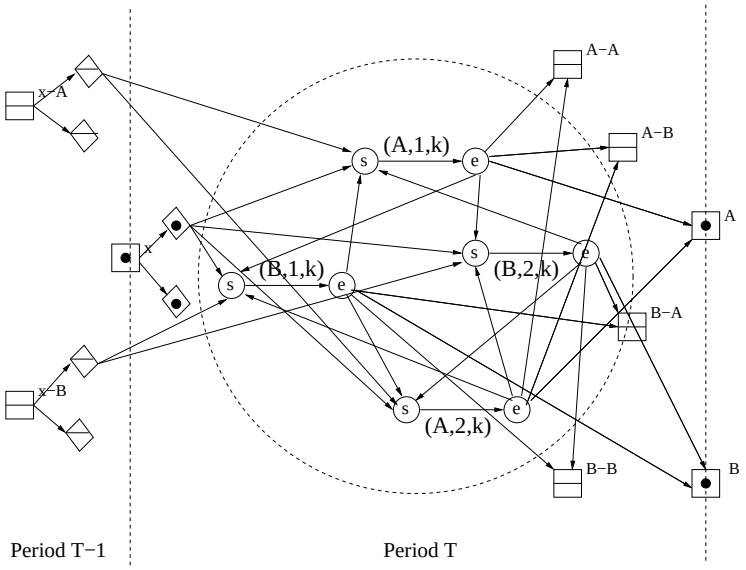


Figure 5.8: A detailed DP network

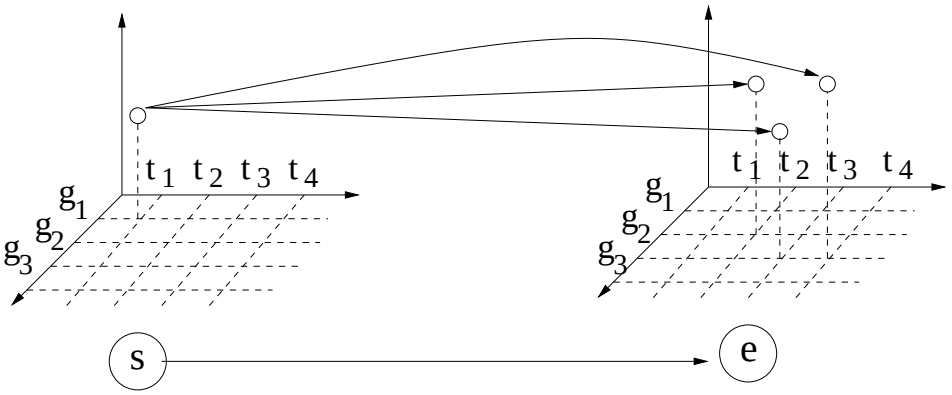


Figure 5.9: Example of a service

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) in Figure 5.8 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 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.

As we mentioned before, ships have three choices after a service in this stochastic model: (1) sail to a start service node and do another service within the same demand scenario part in the network; (2) leave the current node immediately and sail to another start service node in the future time period without any knowledge of future demand situation; (3) wait at the end service node until the demand situation for the future period is revealed, then choose destinations in the future period according to the known demand situation. There are different nodes in Figure 5.8 for each of these actions. For instance if a ship finishes the service for $(A, 1, 1)$, it can go to start service node $(B, 1, 1)$ or $(B, 2, 1)$ and make another service in period 1. This is the 1st action mentioned above. Alternatively it can go to a node $\boxminus^{A-y}$ immediately after the service and go through the corresponding nodes $\diamondsuit$ to a start service node for port y in each of the demand scenario part in next period, which is action (2). In this case, because of lack of demand information for the future period,

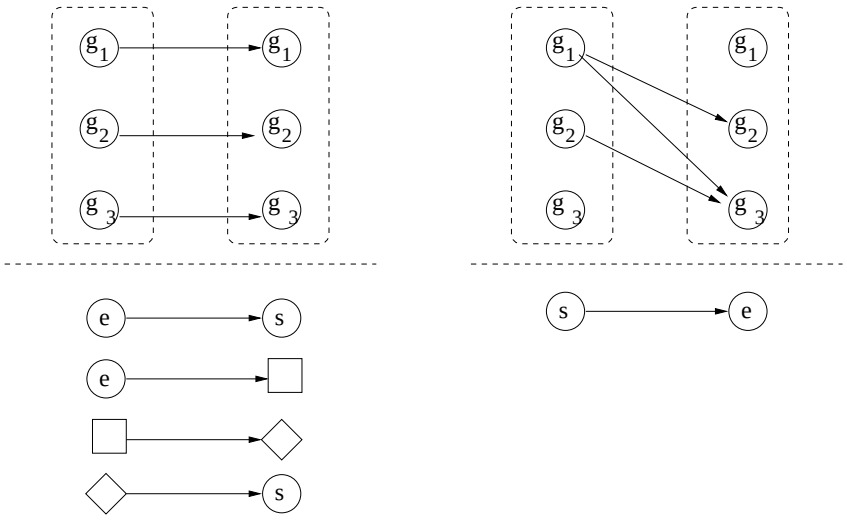


Figure 5.10: Change the network to use different nodes for different quantities

the ship has to decide where to visit next without any information for the future and has to sail to the same physical port in all the parts of future period. It can however go to different port arrivals for the same physical port in different demand scenario parts, because the decision how long to delay before starting the service can be made after the new demand is known. In the example network in Figure 5.8, there are two sum-up node ($\boxminus^{A-y}$ and $\boxminus^{B-y}$, but they will be duplicated for each port y) related to this action. As discussed before, if a ship enters node $\boxminus^{i-j}$, i indicates the physical port where the ship sails from, while j gives the physical port where the ship will sail to in the next period. However, if the ship chooses to wait at physical port i until demand information for the future period is revealed, it goes to $\boxdot^{i}$ node (the third action). Note that the sum-up node $\boxdot^{i}$ is on the boundary of the corresponding period. In this case, it has the freedom to choose the destinations in future parts according to the demand situations.

Once a ship arrives at the final dummy node, the ship finishes its delivery service. The final position for the ship at the end of the planning period can be a supplier or a consumer port. Of course, there is another decision that a ship can make after a service: it can leave the current port visit immediately after the service and sail to some possible central point in sea, and decide where to go once the future demand information is revealed. We discuss this case in Section 6.2 is a diverting model. From the example DP network, we can also see that the end service node $(B, 1, 1)$ is linked with sum-up node $\boxminus^{B-B}$, which indicates that a ship can sail to a port visit of port B in next period from the port visit of the same physical port in current period. This means that a ship's loading or unloading can continue in the following period.

The DP network shown in Figure 5.6 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 5.10 shows how we expand our network using different nodes for different quantities. In the figure, we show the nodes for different quantities of g and the possible links between them above the dotted line. And we show the corresponding arcs below the dotted line. In this example there are three different quantities, and for any arc on the left, the start node and end node should have the same quantity of g because there is no service during a sailing. However, for an arc from Ⓢ to Ⓔ , the Ⓢ node and Ⓔ node usually have different quantities because the quantity on board a ship usually changes during a service. Figure 5.11 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. For example, a ship starts a service at node $(A, 1, 1, g_1)$ with an amount g_1 of commodity on board. It then can finish the service at end service node $(A, 1, 1, g_2)$ having loaded or unloaded the quantity $|g_2 - g_1|$, or at end service node $(A, 1, 1, g_3)$ having loaded or unloaded the quantity $|g_3 - g_1|$. Then it can sail to other start service node for another service. It may then sail from end service node $(A, 1, 1, g_2)$ to start service node $(B, 1, 1, g_2)$ in the network, because the quantity g_2 on board does not change during sailing. 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 5.11 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 all later discussions in the thesis, including the DP recurrence formulation, are 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
Ⓢ $\rightarrow$ Ⓔ	$-d^{Q}_{imk}\lvert g^{E} - g^{S}\rvert - d^{T}_{imk}t^{S}_{imk} - d^{A}_{imk}$	$\lvert g^{E} - g^{S}\rvert / W$
$\diamond \rightarrow$ Ⓢ	$P_k C_{ii'v}$	travel time
Ⓔ $\rightarrow$ Ⓢ	$P_k C_{ii'v}$	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

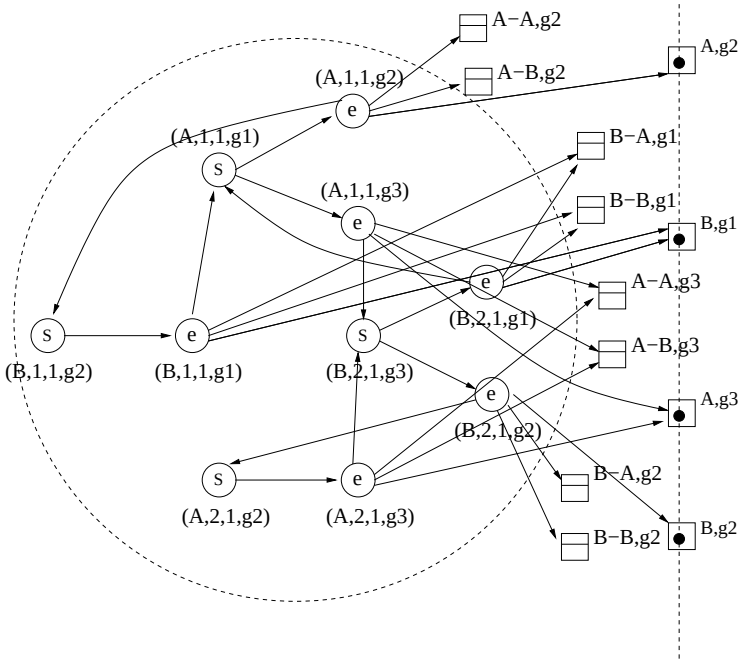


Figure 5.11: Detailed DP network with different end service nodes within a part

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.

5.8 Dynamic Programming Formulation

In this and later sections we assume that there is a separate node for each discrete value of g, and that each node contains the information for all the possible times.

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 ⓢ :

$$f_{lv}(t) = \min_{l':l\to l'}\{f_{l'v}(t+\tilde{T}_{ll'}) + \tilde{C}_{ll'v}\}, \quad t \in [\tilde{A}_{lv}, \tilde{B}_{lv}] \tag{5.20}$$

- For split node $\diamond$:

$$f_{lv}(t) = \min_{l':l\to l'} \min_{\max\{\tilde{A}_{l'v}, t+\tilde{T}_{ll'}\}\le\tau\le\tilde{B}_{l'v}} \{f_{l'v}(\tau) + \tilde{C}_{ll'v}\},\ t \in [\tilde{A}_{lv}, \tilde{B}_{lv}] \tag{5.21}$$

- For end service node ⓔ :

$$\begin{aligned} f_{lv}(t) = \min\{&\min_{l':l\to l'} \min_{\max\{\tilde{A}_{l'v}, t+\tilde{T}_{ll'}\}\le\tau\le\tilde{B}_{l'v}} \{f_{l'v}(\tau) + \tilde{C}_{ll'v}\}, \\ &\min_{l':l\to l'}\{f_{l'v}(t+\tilde{T}_{ll'}) + \tilde{C}_{ll'v}\}\}, \quad t \in [\tilde{A}_{lv}, \tilde{B}_{lv}] \end{aligned} \tag{5.22}$$

$\tilde{A}_{lv}$ 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 5.22, 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:

$$f_{lv}(t) = \sum_{l':l\to l'} f_{l'v}(t), \quad t \in [\tilde{A}_{lv}, \tilde{B}_{lv}] \tag{5.23}$$

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.

5.9 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

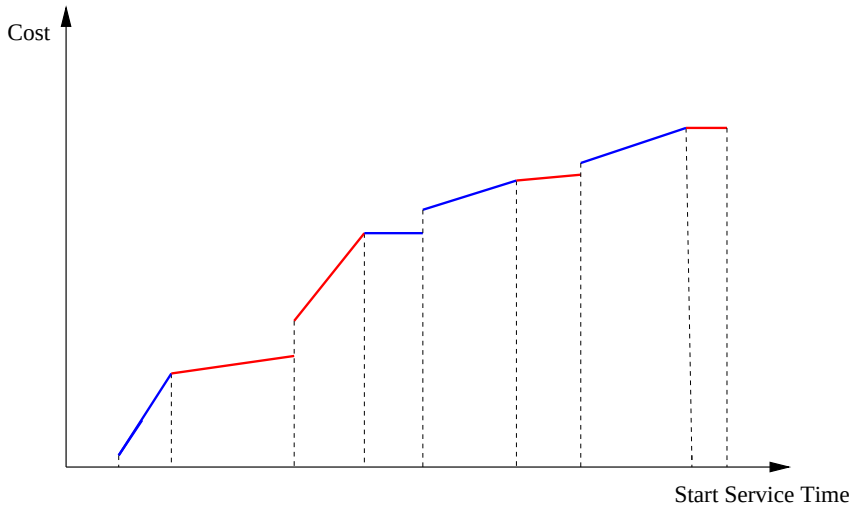


Figure 5.12: Cost function

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 5.12. 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 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. Let $\tilde{f}_{lv}(t)$ be the least

known expected cost from node l at time t to the final node at the start of an iteration of updating. Then in the current iteration, we first use the DP formulation to calculate a temporary cost function $\theta(t)$, and the cost on node l in this iteration is then given by $\min\{\tilde{f}_{lv}(t), \theta(t)\}$. The details of how to solve the subproblem and the examples showing the updating for different types of nodes are given below.

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. Let $\tilde{N}$ be a set of nodes which have been updated in the previous iteration of updating. Figure 5.13– 5.16 show several examples of how to update the cost function for different types of nodes. Figure 5.13 shows how to update the cost function for end service or split node l by start service node l'. The cost and traveling time along the edge $l \rightarrow l'$ is given by $(C_{ll'}, T_{ll'})$. Since delaying before beginning a service is allowed in the model, the cost at time t_1 can be updated by any time greater than the earliest possible start service time at node l' (i.e. $\max\{\tilde{A}_{l'}, t_1 + T_{ll'}\}$), where $\tilde{A}_{l'}$ is the lower bound of the time window. Then the temporary cost of current iteration at time t, $\theta(t)$, can be calculated as following:

$$\theta(t) = \min_{l':l\rightarrow l', l'\in\tilde{N}} \min_{\max\{\tilde{A}_{l'}, t+T_{ll'}\}\leq\tau\leq\tilde{B}_l} \{\tilde{f}_{l'v}(\tau) + C_{ll'}\} \tag{5.24}$$

Since function $\tilde{f}_{l'v}(t)$ is an increasing function, τ can be taken as $\max\{\tilde{A}_{l'}, t + T_{ll'}\}$ when solving the DP. (In chapter 6 we will consider the case where cost of the journey depends on the journey time, and in this case a range of values of τ must be considered.) Then the cost at time t on node l after current iteration of updating can be found as follows:

$$\tilde{f}_{lv}(t) := \min\{\tilde{f}_{lv}(t), \theta(t)\} \tag{5.25}$$

Figure 5.14 shows how to update the cost function of a start service node l by an end service node l'. Because there is no delay allowed within a service, the cost at a time at node l can only be updated by the corresponding time point at node l'. Then the temporary cost at time t of node l can be shown as follows:

$$\theta(t) = \min_{l':l\rightarrow l', l'\in\tilde{N}} \{\tilde{f}_{l'v}(t + T_{ll'}) + C_{ll'}\}, \tag{5.26}$$

And the cost at time t on node l after current iteration of updating can be

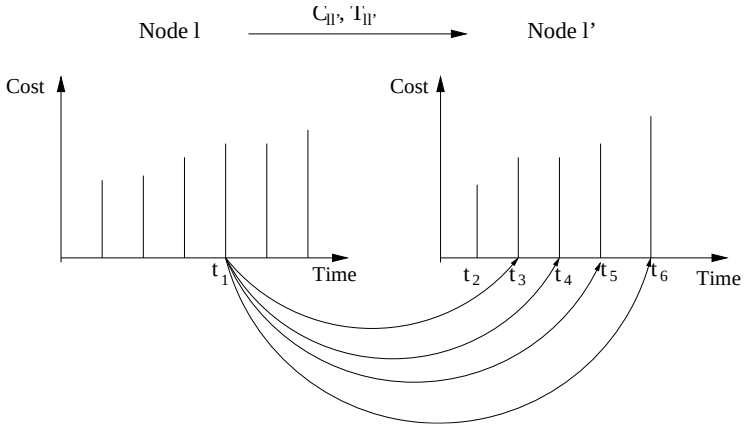


Figure 5.13: Update cost function for end service or split node

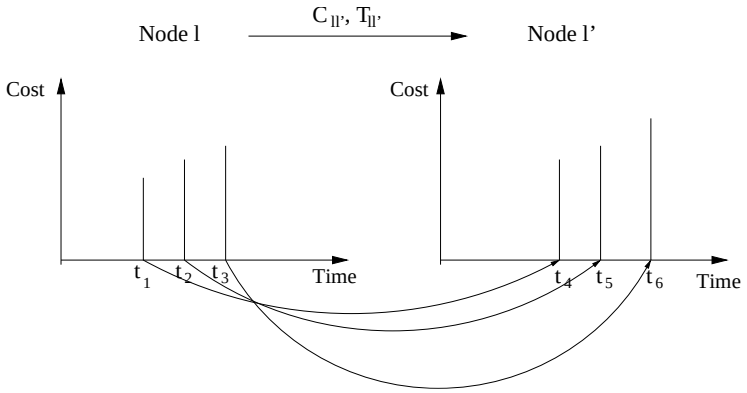


Figure 5.14: Update cost function for start service node

found as following:

$$\tilde{f}_{lv}(t) := \min\{\tilde{f}_{lv}(t), \theta(t)\} \tag{5.27}$$

For sum-up nodes in the network, if any of corresponding split nodes is updated in previous iteration, the cost function of the sum-up node needs to be re-calculated in current iteration of updating. Figure 5.15 shows how to calculate the cost function for sum-up node $\boxminus^{A-B}$ l by split nodes l'_1 and l'_2. Because these nodes have the same time windows, the cost at time t of node l can be easily calculated by the cost of node l'_1 and l'_2 at the same time t:

$$\tilde{f}_{lv}(t) := \min\{\tilde{f}_{lv}(t), \tilde{f}_{l'_1v}(t) + \tilde{f}_{l'_2v}(t)\} \tag{5.28}$$

Figure 5.16 shows how to calculate cost function for sum-up node $\boxdot^{A}$ l by split nodes l'_1 and l'_2. Because the time of these nodes are fixed on the period boundary B_k, the cost at node l can be easily calculated as following:

$$\tilde{f}_{lv}(B_k) := \min\{\tilde{f}_{lv}(B_k), \tilde{f}_{l'_1v}(B_k) + \tilde{f}_{l'_2v}(B_k)\} \tag{5.29}$$

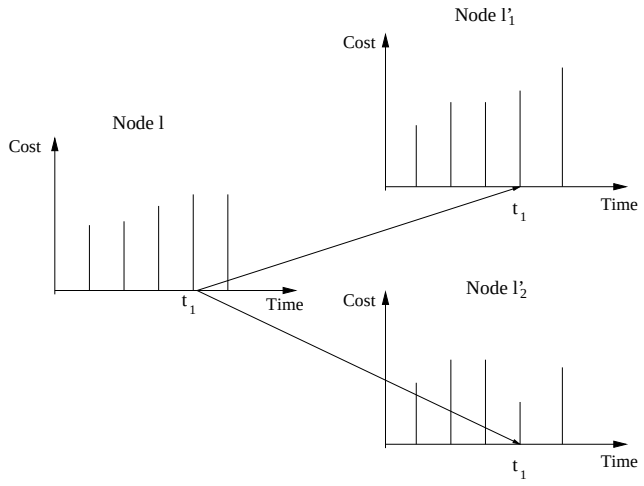


Figure 5.15: Update cost function for sum-up node with the time window before the boundary

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. After the re-ordering, nodes in the network have the following orders:

- A node in a later period has a smaller index than a node in an earlier period.
- A start service node has a bigger index than the end service nodes linked with it.
- A sum-up node has a smaller index than the end service nodes linked with it.
- A split node has a smaller index than the sum-up node linked with it.
- Dummy final node has index 0.
- Dummy start node has the biggest index in the network.

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.

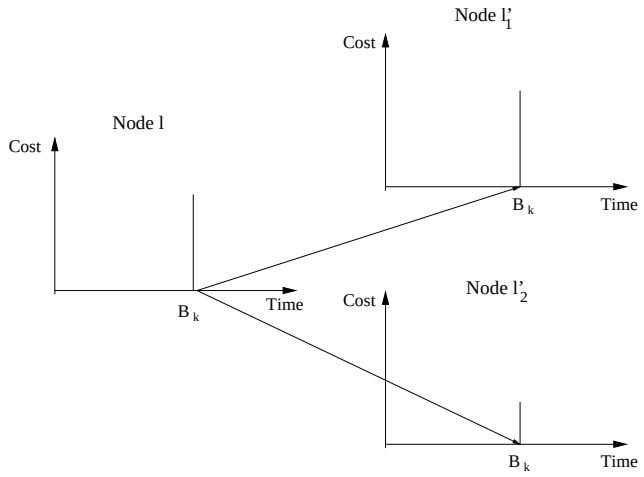


Figure 5.16: Update cost function for sum-up node with the time windows fixed on the boundary

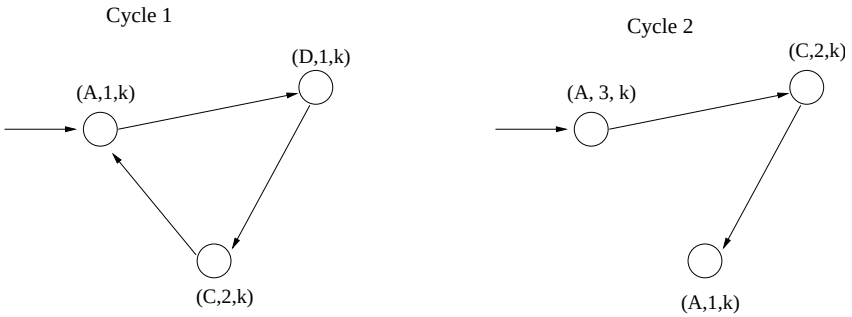


Figure 5.17: Cycles of type A and B

5.10 Cycle Elimination

Because there are several port visits for each port and the order of port visits to different ports is not predetermined, there are a lot of cycles in each demand scenario part of the network. This increases the chance of including illegal cycles in the solution of ship subproblems. There are two types of cycles considered in our problem. Cycle of type A means that a ship returns to some port visit which occurred for that ship before, while cycle of type B means that the ship route contains a port visit (i, m_1, k) and a later port visit (i, m_2, k) where $m_2 \leq m_1$. Such a path cannot logically occur. We show the examples of these two situations in Figure 5.17. We shall refer to both of these cases as cycles even thought the type B may not include a real cycle.

As mentioned in Section 5.4 when there is a cycle of Type A, 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,

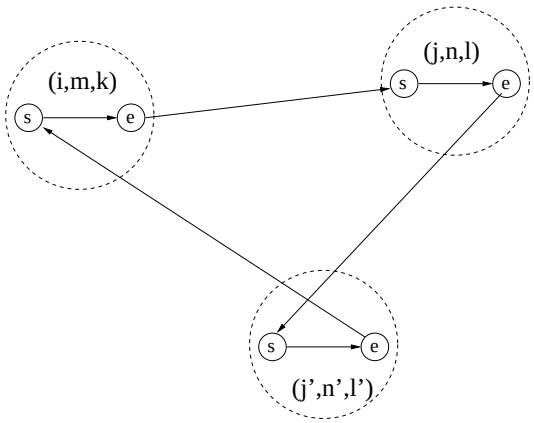


Figure 5.18: A type A cycle in our DP network

so bounds allowing them are still valid. However, a solution with cycles is not logically feasible. Because allowing cycles gives a relaxation of the true situation, 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 Branch-and-Bound 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 2006 and Irnich and Desaulniers 2004. However, it is not easy to avoid all the cycles with different length 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 Figure 5.18 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.

In our problem, cycles only occur in each scenario part of the DP network, and there is no cycle crossing the period boundary. 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 Irnich and Villeneuve 2006, 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 A and B 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. Let $\tilde{f}_{lv}^{(1)}(t)$ and $\tilde{f}_{lv}^{(2)}(t)$

be, respectively, the current stored least and second least expected cost along allowed paths from node l at time t to the final node in the network without 2-cycle. $\tilde{P}^{(1)}_{lvt}$ and $\tilde{P}^{(2)}_{lvt}$ are the stored partial path (the first port visit after the current node l) on node l at time point t for the best and second best solution respectively. Assume that PV_l is the port visit associated with node l. Let θ be the temporary cost calculated in the current iteration of updating and P the corresponding partial path at time t. During the updating, if θ is calculated, we will use this cost to update the current stored best and second best list.

In an iteration of updating we check each node in the DP network, consider the every outgoing arc from the node, and update the cost function of the node by using the end node of the arc. Let l be the start node of an arc and l' the end node. If node l is an end service node ⓔ and node l' is a start service node ⓢ, we need to check, for every time point, whether the best solution or second best solution on node l' causes a 2-cycle. If the best solution of node l' at time τ causes a 2-cycle, then second best solution will be considered to calculate the temporary cost θ. This is shown in the following Pseudo-code:

begin: ⓔ → ⓢ
  for ($t \in [\tilde{A}_{lv}, \tilde{B}_{lv}]$) **do**
    $\theta = \infty,\ P = PV_{l'}$;
    for ($\tau \in [\tilde{A}_{l'v}, \tilde{B}_{l'v}]$) **do**
      if ($\theta > \tilde{f}^{(1)}_{l'v}(\tau) + C_{ll'}$) **do**
        if ($P^{(1)}_{l'v\tau}$ not same or earlier than current port visit) **do**
          $\theta = \tilde{f}^{(1)}_{l'v}(\tau) + C_{ll'}$; (no 2-cycle with best)
        else
          if ($\theta > \tilde{f}^{(2)}_{l'v}(\tau) + C_{ll'}$) **do**
            if ($P^{(2)}_{l'v\tau}$ not same or earlier than current port visit) **do**
              $\theta = \tilde{f}^{(2)}_{l'v}(\tau) + C_{ll'}$; (no 2-cycle with 2nd best)
            end-if
          end-if
        end-if
      end-if
    end-for
    checkUpdateList($\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt}$));
  end-for
end

If node l is a start service node and node l' is an end service node, we do not need to check 2-cycles when updating the cost function. This is because that 2-cycles will be avoided automatically for start service nodes if they are eliminated already on the corresponding end service nodes. For updating cost

functions of start service nodes, we need to use both best and second best to calculate temporary costs at each time point. See the following Pseudo-code for details:

begin: ⓢ → ⓔ
  for $(t \in [\tilde{A}_{lv}, \tilde{B}_{lv}])$ **do**
    $\theta = \tilde{f}^{(1)}_{l'v}(t + \tilde{T}_{ll'}) + C_{ll'}$; $P = P^{(1)}_{l'v,t+\tilde{T}_{ll'}}$;
    checkUpdateList$(\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt})$;
    $\theta = \tilde{f}^{(2)}_{l'v}(t + \tilde{T}_{ll'}) + C_{ll'}$; $P = P^{(2)}_{l'v,t+\tilde{T}_{ll'}}$;
    checkUpdateList$(\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt})$;
  end-for
end

If node l is a $\diamond$ node and node l' is a start service node. We need to consider the possible delays of journey, but no cycles need to be checked since there is no cycle cross the period boundaries. See the following Pseudo-code for details:

begin: $\diamond \rightarrow$ ⓢ
  for $(t \in [\tilde{A}_{lv}, \tilde{B}_{lv}])$ **do**
    $\theta = \infty$;
    for $(\tau \in [\tilde{A}_{l'v}, \tilde{B}_{l'v}])$ **do**
      if $(\theta > \tilde{f}^{(1)}_{l'v}(\tau) + C_{ll'})$ **do**
        $\theta = \tilde{f}^{(1)}_{l'v}(\tau) + C_{ll'}$;
        $P = PV_{l'}$;
      end-if
    end-for
    checkUpdateList$(\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt}))$;
  end-for
end

If node l is an end service node and node l' is sum-up node $\boxed{\bullet}^i$, we do not need to check 2-cycles. Let B be the period boundary related to $\boxed{\bullet}^i$, and the details about updating are given below:

begin: ⓔ → $\boxed{\bullet}^i$
  for $(t \in [\tilde{A}_{lv}, \tilde{B}_{lv}])$ **do**
    $\theta = \tilde{f}^{(1)}_{l'v}(B)$;
    $P = P^{(1)}_{l'vB}$;
    checkUpdateList$(\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt}))$;
  end-for

end

For all other arcs the start nodes can be easily updated by end nodes as following:

begin: $\textcircled{e} \rightarrow \boxminus^{i-j}$, $\square \rightarrow \diamond$
  for $(t \in [\tilde{A}_{lv}, \tilde{B}_{lv}])$ **do**
    $\theta = \tilde{f}^{(1)}_{l'v}(t)$;
    $P = P^{(1)}_{l'vt}$;
    checkUpdateList$(\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt}))$;
  end-for
end

In above Pseudo-codes, details of subroutine checkUpdateList$(\theta, P, \tilde{f}^{(1)}_{lv}(t), P^{(1)}_{lvt}, \tilde{f}^{(2)}_{lv}(t), P^{(2)}_{lvt})$ to update the best and second best solutions. The details of this subroutine are given below. Assume that the port visit stored in P is (i_T, m_T, k), the port visit stored in P^1_{lvt} is (i_{B1}, m_{B1}, k) and the port visit stored in P^2_{lvt} is (i_{B2}, m_{B2}, k). If $(i_T, m_T, k) = (i_{B1}, m_{B1}, k)$, we update the list as shown in the following table:

	Situation	Updates
1	$\theta < \tilde{f}^{(1)}_{lv}(t)$	$\tilde{f}^{(1)}_{lv}(t) = \theta$, $\tilde{P}^1_{lvt} = P$
2	$\tilde{f}^{(1)}_{lv}(t) \leq \theta < \tilde{f}^{(2)}_{lv}(t)$	no update
3	$\theta > \tilde{f}^{(2)}_{lv}(t)$	no update

Table 5.1: Update best and 2nd best list 1

In table 5.1 situation 1 says that if the corresponding partial paths for the temporary solution and the best solution are the same and temporary cost is better than the best cost, we can only update the best solution by the temporary one but nothing changed for the second best solution. In situation 2 and 3, nothing will be changed to the best and second best list. If $i_T \neq i_{B1}$, we update the list as shown in the following table:

	Situation	Updates
1	$\theta < \tilde{f}^{(1)}_{lv}(t)$	$\tilde{f}^{(2)}_{lv}(t) = \tilde{f}^{(1)}_{lv}(t)$, $\tilde{f}^{(1)}_{lv}(t) = \theta$ $\tilde{P}^2_{lvt} = \tilde{P}^1_{lvt}$, $\tilde{P}^1_{lvt} = P$
2	$\tilde{f}^{(1)}_{lv}(t) \leq \theta < \tilde{f}^{(2)}_{lv}(t)$	$\tilde{f}^{(2)}_{lv}(t) = \theta$, $\tilde{P}^2_{lvt} = P$
3	$\theta > \tilde{f}^{(2)}_{lv}(t)$	no update

Table 5.2: Update best and 2nd best list 2

In above table, situation 1 says that if the temporary cost is better the current stored best cost and the corresponding partial paths are not the same, we can update the second best cost by the current stored best cost and update the best cost by the temporary cost. In situation 2, if the temporary cost is better than the second best cost but worse that the best cost and corresponding partial paths for the temporary solution and the best solution are different, we update the second best solution. If $i_T = i_{B1}$, we need to consider different situations about m_T and m_{B1} when updating the list, which is shown in the following table:

	Situation	$m_T > m_{B1}$	$m_T < m_{B1}$
1	$\theta < \tilde{f}_{lv}^{(1)}(t)$	$\tilde{f}_{lv}^{1}(t) = \theta$ $P_{lvt}^{1} = P$	$\tilde{f}_{lv}^{(2)}(t) = \tilde{f}_{lv}^{(1)}(t)$, $\tilde{f}_{lv}^{(1)}(t) = \theta$ $\tilde{P}_{lvt}^{2} = \tilde{P}_{lvt}^{1}$, $\tilde{P}_{lvt}^{1} = P$
2	$\tilde{f}_{lv}^{(1)}(t) \leq \theta < \tilde{f}_{lv}^{(2)}(t)$	$\tilde{f}_{lv}^{(2)}(t) = \theta$ $\tilde{P}_{lvt}^{2} = P$	no update
3	$\theta > \tilde{f}_{lv}^{(2)}(t)$	no update	no update

Table 5.3: Update best and 2nd best list 3

In order to avoid type B cycle, we need to keep m-value of P_{lvt}^{1} less than the m-value of P_{lvt}^{2}. If $i_T = i_{B1} = i_{B2}$, we may get some complications when $m_T > m_{B1}$ and $\theta < \tilde{f}_{lv}^{(1)}(t)$. In this case, if $m_T > m_{B2}$, then we eliminate the second best from the list, and just leave the best. Then during the updating, we choose the best path even though it results a cycle.

5.11 Branch and Bound Algorithm

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. In this thesis, 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

$\bar{t}_{imk} = 1/K \sum_{y=1..K} t^{Sy}_{imk}$ 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 $x_{imkjnlv} = \sum_{s \in R_v; (i,m,k) \rightarrow (j,n,l) \in E_s} x_{sv}$. 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.

5.12 Computational Experiments

To test the models and solution methods developed in this thesis, 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.

5.12.1 Implementation

Before giving the test examples and computational results, we first give the pseudo-code and major structure of the implementation here. We first illustrate the branch-and-bound algorithm as below. The networks are built before solving the problem and input as data. A Depth First Branch-and-Bound algorithm is used to find an integer solution. At each branch-and-bound node, we call the subroutine to solve the restricted master problem by a column generation method.

Pseudo-code for B&B Algorithm:
begin
Input network structure and initial columns;
$OPTval = \infty$;
Build matrix for solving;
Create and store B&B node {1}, Tree = {1};
while Tree is not empty **do**
Get last stored B&B node $\{J\}$, Tree = Tree/$\{J\}$;
Update networks;
Check matrix, delete infeasible columns;
Generate artificial columns for the node problem;
call nodeProblem();

```
        if (artificial column is chosen in solution) do
            cutoff = True (Infeasible problem);
        end-if;
        if (integer solution found && cutoff is False) do
            if (LPval < OPTval) do
                OPTval = LPval;
            end-if;
        else if (cutoff is False) do
            Choose branching variable and do branching;
            Create child nodes and store B&B nodes in B&B tree;
        end-if;
    end-while;
    if (OPTval < ∞) do
        Report optimal integer solution;
    end-if;
end;
```

The branch-and-bound algorithm is quite simple. We call subroutine "nodeProblem()" at every branch-and-bound node. After solving the node problem, boolean variable $cutoff$ indicates whether the node is cutoff or not. If not cutoff, we choose the branching variable and expand the branch-and-bound tree. We report each feasible solution found and continue until there is no node in the tree. The pseudo-code of the node problem is shown below:

```
Pseudo-code for nodeProblem():
begin
    cutoff = False; updated = True;
    while (updated is True) do
        updated = False;
        Solve the master problem by calling CPLEX;
        if (# of simplex iteration > 0) do
            Get dual variables and update edge costs in the DP networks;
            call shipSubProblem(), get reduced cost reducedC_v, v ∈ V;
            if (LPval + Σ_{v∈V} reducedC_v ≥ OPTval) do
                cutoff = True;
            else
                for all ships do
                    if (reducedC_v < 0) do
                        Add the column into master problem;
                        updated = True;
                    end-if;
                end-for;
            end-if;
```

```
        end-if;
    end-while;
end;
```

In the node problem, we solve the master problem to get the dual variables. Then the new edge costs of DP networks are updated by using these dual variables. New columns with least reduced costs are generated by calling ship subproblems. If the LP solution of the master problem plus the least reduced costs is greater than the current optimal integer solution, then we cannot find the better integer solution on this branch, and the branch-and-bound node is cutoff. Otherwise, the negative least reduced cost columns are added into the master problem, and the master problem is solved again. This process continues iteratively until there is no negative reduced cost column can be generated from the ship subproblems.

We solve the ship subproblems by labeling method. When we do branching, some port visits or arcs in the DP network may be removed from the network. We use the array variable *nodelive*[ ] and *edgelive*[ ] in the subproblem to indicate whether a node or an edge has been removed or not. Before updating a node cost function, we first check whether there is a 2-cycle. We go through all the nodes in the network iteratively to update the cost function of each node until no more updates can be done. Since cycles are only exist within a demand scenario part of the network, we only consider 2-cycle elimination for start and end service nodes when updating the cost function through the network. Other nodes are updated without considering the cycle elimination. The detailed algorithm about updating costs by considering 2-cycle elimination has been given in Section 5.10.

In next section, we give the details of all test examples and corresponding computational results.

5.12.2 Examples and Results

Computational results are reported in this section. Table 5.4 gives the characteristics of each test problem.

In table 5.4, $a1$ 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 in Section 5.4. 'Scenario Nodes', 'Planning Periods' and 'Branches' columns give the structure of the scenario tree. For example, in example $g1$, there are 13

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 5.4: Example Information

scenario tree nodes, 3 time periods and 3 branches each period in the scenario tree, which indicates a scenario tree as shown in Figure 5.19.

The transportation networks of these examples are shown in Figure 5.20. There is a set of ports in each network, some of which are consumer ports and others are supply ports. There are arcs between the supply ports and the consumer ports, as well as between the two close consumer ports, but there is no arc between any two supply ports.

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 (i, m, k),

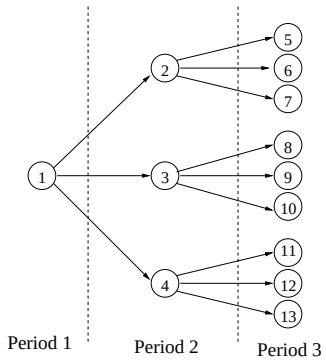


Figure 5.19: Scenario tree of Ex. g1

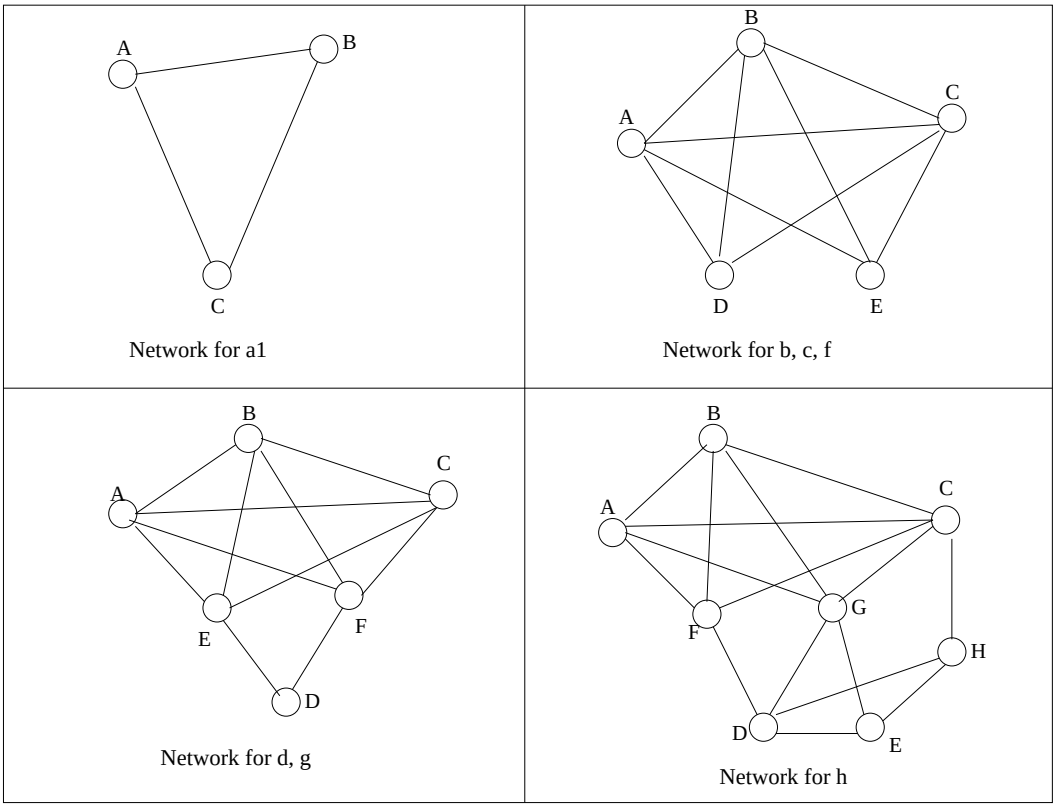


Figure 5.20: Physical port layout of test problems

the start service time of the port visit should be within a time window. In all the examples here, we set the time window for each port visit (i, m, k) to the time period including 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 5.7. 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 5.5. The table also shows the number of (i, m, k) combinations and the number of constraints in the master problem.

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 5.5: DP and Master Problem Dimensions

It can be seen from Table 5.5 that there is a large number of nodes and edges in each ship subproblem. When solving the DP for these subproblems, we need to go through all of these nodes and edges repeatedly to update cost functions on each node in the network. As a result the major solving time in all of the examples is used to solve the ship subproblems. The computational results are shown in Table 5.6.

Table 5.6 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

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 5.6: Computational Results without Tolerance

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 major solving time in each example is used to solve the ship subproblems, and Table 5.6 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.

When we consider the tolerance of 5%, a possible improvement of solving performance can be expected with possibly worse objective values. Here the tolerance is defined as following:

$$\frac{OPTval - LB^{min}}{IPSol} \leq 5\% \tag{5.30}$$

This means that the branch-and-bound algorithm will be stopped if the current minimum lower bound is within a 5% range of the current stored IP solution. Table 5.7 shows the comparisons between the results with the tolerance of 5% and the results without using tolerance. The figures shown in the table give the percentage improvement on number of branch-and-bound nodes and number of columns generated, as well as the percentage of how much the objective values get worse by using the tolerance.

EX	B&B nodes	Columns	Objectives
b1	0 %	-0.8 %	0 %
b2	-6.78 %	-9.03 %	3.03 %
b3	-19.18 %	-21.79 %	0 %
c1	-7.41 %	-12.65 %	0 %
c2	0 %	0 %	0 %
c3	0 %	0 %	0 %
d1	-26.8 %	-19.1 %	0 %
d2	-39.3 %	-22.1 %	1.3 %
d3	-20.8 %	-16.5 %	0.5 %
f1	-30.9 %	-33.2 %	0.7 %
f2	-45.65 %	-34.67 %	0 %
g1	-15.9 %	-17.9 %	1.0 %
g2	-34.13 %	-40.21 %	3.86 %
g3	-38.73 %	-31.39 %	0 %
h1	-19.07 %	-12.54 %	4.3 %
h2	-20.1 %	-30.17 %	4.7 %

Table 5.7: Comparison between the results with and without tolerance

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 5.21 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 5.21 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

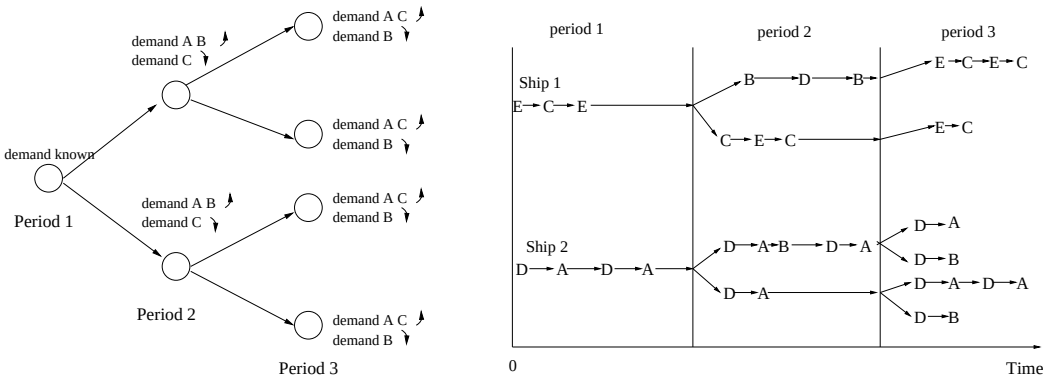


Figure 5.21: Solution c1

because all of the demands are satisfied in the case so that there is no need to travel any further.

Figure 5.22 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 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.

5.13 Summary and Discussion

This chapter presents the decomposition formulation model for the stochastic ship routing problem, and the branch-and-price algorithm is used to solve the problem. A column generation method is used to solve the node problem on each branch-and-bound node. On each node, a master problem is built, which has the similar format as the set partitioning formulation, but together with a group of inventory constraints. The master problem is solved by linear programming by calling CPLEX 10.0. A set of subproblems is also built, one for each ship, to generate promising columns for the master problem. The subproblems are formulated as a dynamic programming problem and solved by labeling method introduced in Desrochers and Soumis 1988a, Desrochers and Soumis 1988b and Desrosiers et al. 1995.

In order to test the model, a group of examples are generated and the computational experience shows that the solving time increases rapidly with problem size and is also influenced by initial storage levels. The major solution time is used to solve the subproblems. Because the subproblems are

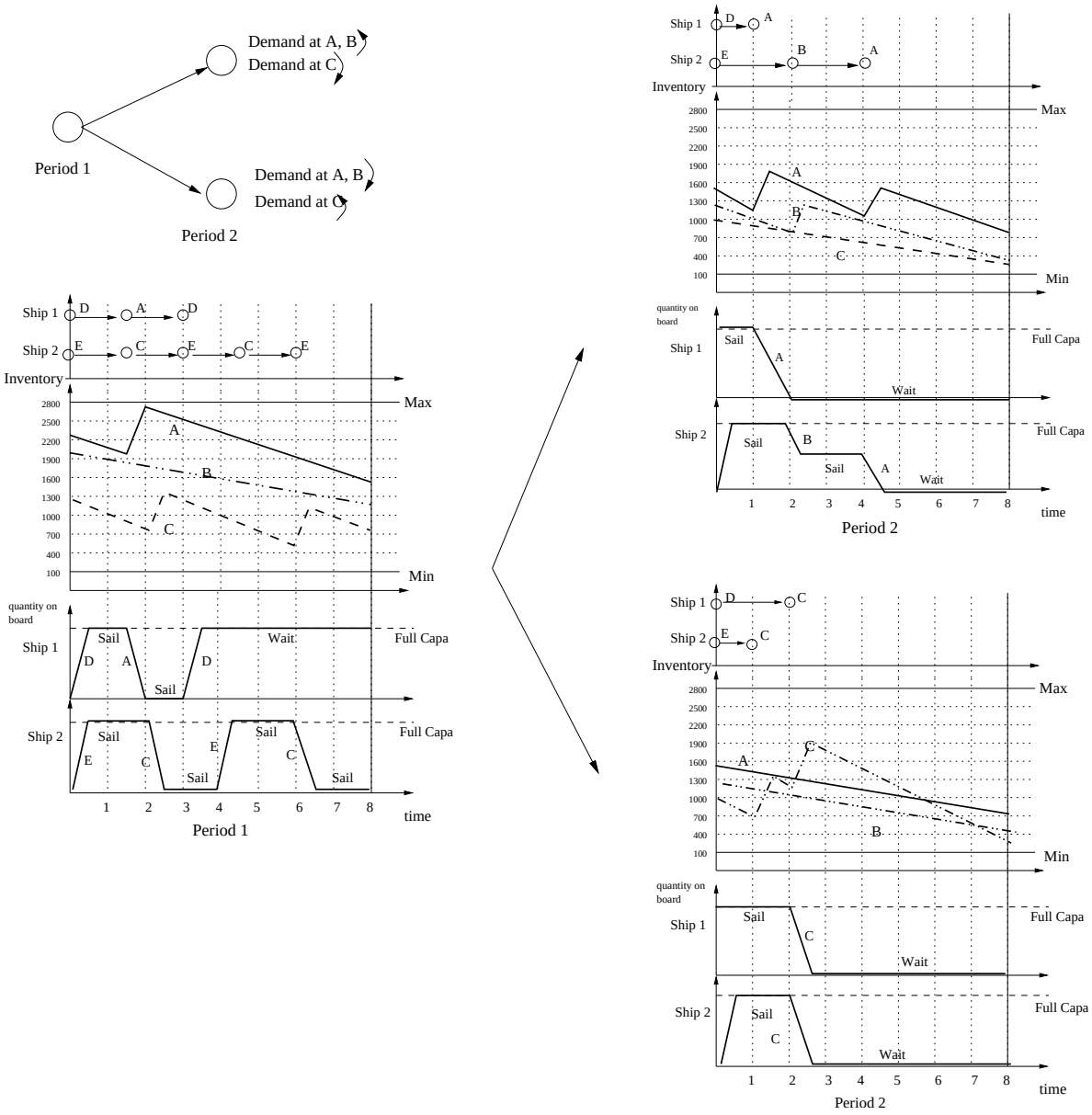


Figure 5.22: Solution Example b1

independent each other, they are solved in parallel using OpenMP. Because of the large size of the stochastic model, the stochastic ship routing problem is a very hard combinational problem, and problems of a very large size can still not be solved, however the method developed in this chapter are much more efficient than the MIP formulation given in Chapter 4.

Chapter 6

Extension Models

In Chapter 5, the decomposition model of the stochastic ship routing problem is presented and its solution by Branch-and-Price algorithm is described. Based on this model, several extensions of the original model are described in this chapter. These extended models deal with more realistic situations in the real world shipping industry. Rest of the chapter has the following sections. A variable speed model is given in Section 6.1. This model allows ships to sail at different speeds on different legs of their routes. Section 6.2 presents a divert model, which allows ships to divert during sailing. In the stochastic model, demand information is not known until the beginning of a period, but the traveling decisions may have already been made before the information is revealed, so once the demand information is known, ships may need to change their destinations. The divert model presented in this chapter allows ships to sail into some central position between the possible destinations and decide the direction of sailing after knowing the future demand. In Section 6.3, a model with bonus for additional work during idle time is introduced. Some ideas about building the model are discussed but not implemented.

6.1 Variable Speed Model

6.1.1 Introduction

In most of the literature about ship routing problems, the ship speed is assumed to be constant and the corresponding traveling time between any two ports in the transportation network is fixed. In the real shipping industry, each ship is designed for a particular speed when it is built, but it is able to sail at other speeds. In Chapter 5, ships always sailed at their speed, though they were able to delay their arrival by an arbitrary amount. However the cost of a journey was always the same and the minimum time for a journey was fixed by the ship's speed. However, if the ship speed can be varied the schedule may be improved by sailing at different speeds between different ports. Hence

the variable speed model is a more general case of the ship routing problem and the fixed speed model is a special case of the variable speed model.

As a ship's speed increases from its design speed the traveling time to cover a fixed distance decreases but the cost of fuel increases. On the other hand, if a ship's speed decreases from its design speed the traveling time increases and the cost of fuel decreases. The fuel cost to travel unit distance, $\tilde{C}_v$, in time $\hat{t}_v$ is usually assumed to be of the following formula:

$$\tilde{C}_v = \frac{\beta}{\hat{t}_v^3} \tag{6.1}$$

where β is a constant. The formula says that the traveling cost is inversely proportional to the cube of the traveling time. For two ports a distance L_{ij}, the traveling time between the ports is $t_{ijv} = L_{ij}\hat{t}_v$. Then the traveling cost from port i to port j can be calculated as following:

$$\begin{aligned} \tilde{C}_{ijv} &= L_{ij}\tilde{C}_v \\ &= L_{ij}\frac{\beta}{\hat{t}_v^3} \\ &= L_{ij}\frac{\beta}{(t_{ijv}/L_{ij})^3} \\ &= \frac{\beta L_{ij}^4}{t_{ijv}^3} \end{aligned} \tag{6.2}$$

In practise, we add an additional item into above formula to represent any additional cost caused by operations.

$$\tilde{C}_{ijv} = \tilde{\theta} + \frac{\beta L_{ij}^4}{t_{ijv}^3} \tag{6.3}$$

An example of the cost function over traveling time is illustrated in Figure 6.1. This is a nonlinear decreasing function which makes the problem nonlinear.

In the variable speed model, we again use a set of discrete small time slots, which are much shorter compared with the whole planning period. By using these small time slots we can handle the variable speed situations well. Difference between two different speeds can be represented by the traveling time between two ports covering different number of small time slots. Furthermore, the master problem may take the average of more than one columns as the solution, and the local optimization model can be used to generate the optimal speed (or arriving time) in our problem. Therefore, our

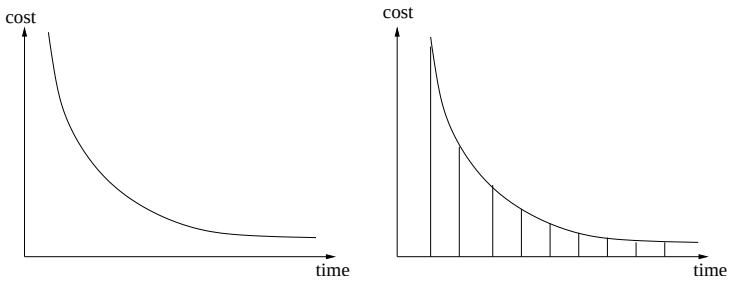


Figure 6.1: Discretised cost function

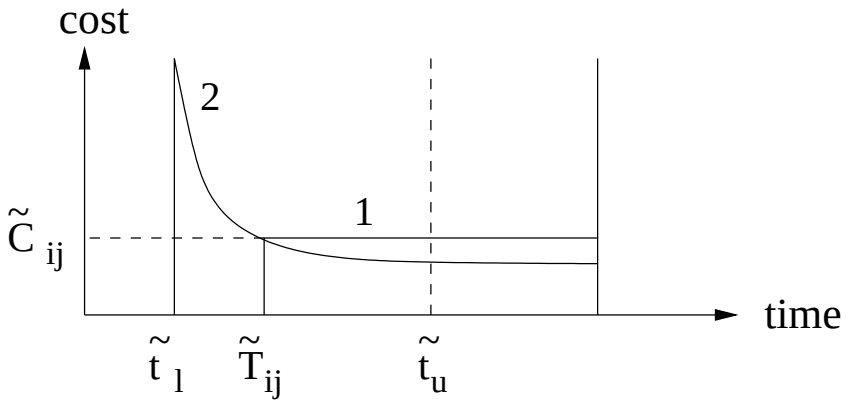


Figure 6.2: Comparison of traveling cost function between original (1) and variable speed model (2)

solution of the variable speed model can be different from the discrete points and mainly reflect the optimal speed.

In the rest of the section, Section 6.1.2 shows the changes made to the original model to build the variable speed model. Section 6.1.3 gives details of a mathematical formulation of a local optimization model. The computational results are given in Section 6.1.4.

6.1.2 Changes to the Original Model

The original decomposition model consists of two main parts, the master problem and a set of subproblems. To allow for the variable speed in the model, some changes have to be made to the subproblems, but there is no change to the master problem. The major difference between the original DP and the DP for the variable speed model is that different traveling cost functions are used on the traveling edges in the DP network. In the original DP, the traveling cost is constant, while in the variable speed model the cost function is a decreasing convex function (or a piece-wise linear function as an approximation). See Figure 6.2 for example.

In Figure 6.2, curve 1 shows the edge costs in the original DP network.

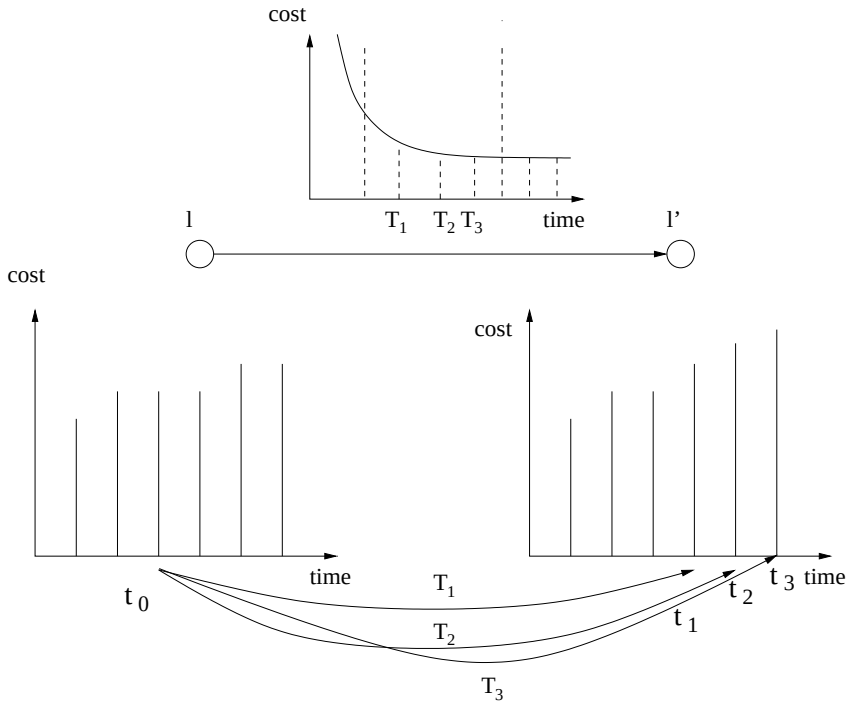


Figure 6.3: Update cost function in variable speed model

Because the traveling cost is fixed, the cost function is a horizontal line. In this case, the minimum traveling time between port i and j for a ship is $\tilde{T}_{ij}$ with fixed traveling cost $\tilde{C}_{ij}$. Any transition time greater than $\tilde{T}_{ij}$ means that the ship sails for $\tilde{T}_{ij}$ and then waits at port j without any waiting cost. Curve 2 shows the traveling cost function in the variable speed model. There are lower and upper bound of traveling time which correspond to the highest and lowest sailing speed of the ship. After the highest traveling time $\tilde{t}_u$, the cost function goes flat because ships can delay without any cost after sailing with the corresponding lowest speed between two ports.

The computational experience of the original model showed that the major solving time is spent on solving the ship subproblems. To prevent the solution times increasing significantly we discretize the traveling times in the same way as for the constant speed case. However there is still more work than in the constant speed case because we can no longer use only the minimum travel time but in the variable speed case all times between the slowest and highest mush be examined.

When updating costs in the variable speed model, we consider each possible traveling time between two nodes in the DP network and get the corresponding traveling cost by checking the cost from the cost function. Figure 6.3 illustrates the updating process.

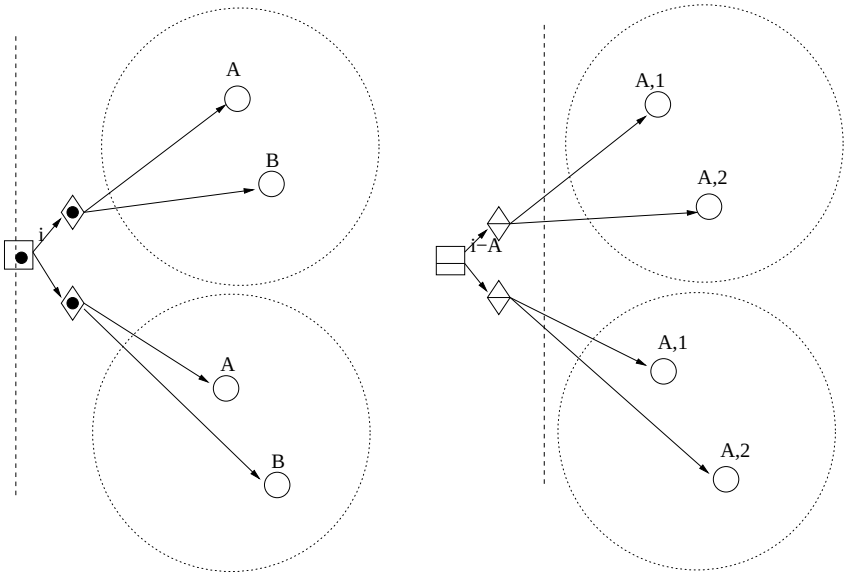


Figure 6.4: Decisions of crossing boundary in variable speed model

In the example, we calculate the cost of node l at time t_0 using the cost function of node l'. From Figure 6.3, the transition time between t_0 to t_1, t_0 to t_2 and t_0 to t_3 are T_1, T_2 and T_3 respectively. Then according to the cost function on the edge $l \rightarrow l'$, the traveling costs are $\tilde{C}_{ll'}(T_1)$, $\tilde{C}_{ll'}(T_2)$ and $\tilde{C}_{ll'}(T_3)$. Then the temporary cost at time t_0 can be calculated as following:

$$\begin{aligned} \theta(t_0) \quad &= \min\{\tilde{f}_{l'}(t_1) + \tilde{C}_{ll'}(T_1), \tilde{f}_{l'}(t_2) + \tilde{C}_{ll'}(T_2), \\ &\quad \tilde{f}_{l'}(t_3) + \tilde{C}_{ll'}(T_3), ...\} \end{aligned} \tag{6.4}$$

The use of discrete traveling times in a strictly convex cost function can lead to discretization error. We may be not able to find the local optimal solution when the cost function is nonlinear even when taking the average of discrete values. This can be overcome by using a local optimization to find the solution. This will be discussed in detail in Section 6.1.3.

As discussed in Chapter 5, there are two types of sum-up nodes ($\boxminus^{i-j}$ and $\boxdot^{i}$) in the DP network. Because the demand information is revealed when a ship arrives at a $\boxdot^{i}$ node, decisions made on the corresponding $\diamond$ nodes linked with $\boxdot^{i}$ are: (1) which physical port to visit in the next period and (2) when to start the service at the destination port. In this case, ships can choose their speeds along the outgoing arcs from $\diamondsuit$ nodes. See the left part of Figure 6.4 for example.

However, in the example shown in the right part of Figure 6.4 since no future information can be known when a ship arrives at $\boxminus^{i-A}$ node, decisions have been made on the corresponding $\diamondsuit$ nodes without any knowledge of

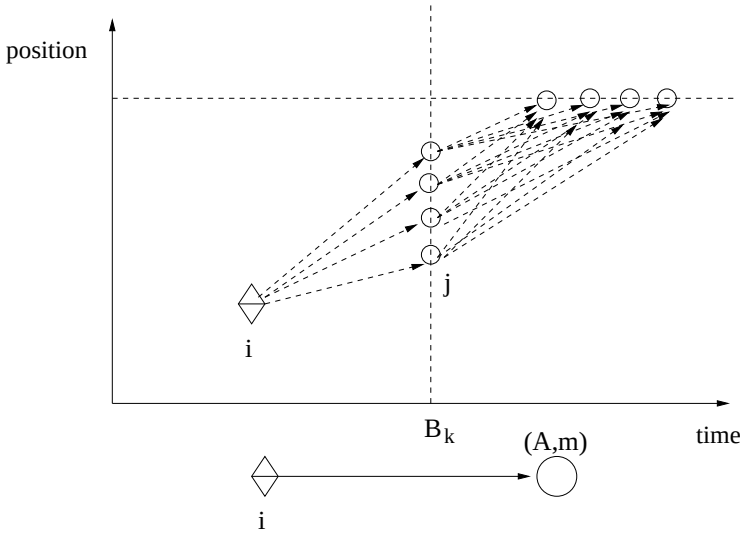


Figure 6.5: Introducing dummy points on the arcs crossing period boundary

future demands. Therefore, the ship has to take the same action on both ◇ nodes until the boundary of the period. The same action here means that the ship has to choose the same destination port in all scenarios, and it has to sail at the same speed before the end of current period. The ship has to choose the same speed on the outgoing arcs from both ◇ nodes to port A before the period boundary, but after the boundary, the ship can choose different speeds according to the known demand information in different scenario demand part. This separates the journey from port i to A into two segments. Because we do not allow ships to change their speeds during a single journey, this situation does not fit with our model.

When we implement the variable speed model, we do not allow a choice of speeds when a ship crosses a period boundary. We make an assumption that every time a ship crosses the period boundary via a $\boxminus^{i-j}$ node, the ship can only sail at a single fixed speed, and cannot change its sailing speed at the period boundary. Doing this makes the solutions of the variable speed model sub-optimal instead of optimal. The computational results given in later section are based on this assumption.

To avoid this sub-optimal behaviour we could introduce a set of dummy points along the arcs from ◇ nodes to port A in the right part of Figure 6.4. See Figure 6.5 for example. When a ship leaves port i, it can choose one speed to sail towards port A, and arrive at dummy point j at time B_k (the end boundary of current period). Then along the journey from point j to port A, the ship can change to another speed and sail to port A according to the different demand information. By considering this situation we can find the optimal solution from the variable speed model. This case with the dummy points on the arcs crossing the time period boundary is a special case of the

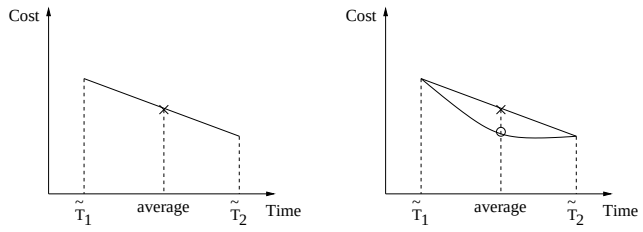


Figure 6.6: Average and optimal

diverting model which will be introduced in Section 6.2 later.

6.1.3 Local Optimization

When we solve the variable speed model, we discretise the traveling time in the cost function of each edge in the DP network. So the ship subproblem can only generate the tree of routes with the discretised traveling time. This however does not mean that we can only find solution with discrete value of traveling time. The master problem can choose several columns with the same physical route but different times and use average of these columns as the solution. The linear combination of these columns gives a solution with traveling times which may be not on the discrete grid. Then a question caused by this is whether or not the average gives the optimal solution.

Obviously, if the cost function between the traveling times is a straight line, the average of the columns with the traveling times gives the correct cost. However, if the cost function is strictly convex between these times, the average of the two columns has higher cost than the correct one. This is shown in figure 6.6. We can clearly see that there is a gap between the average value and the real value in the strictly convex case.

In order to avoid this error, we can do a local optimization in which given a set of trees of routes for each ship, the optimal timing and load quantities can be found by solving a NLP. Every time the master problem is solved, we need to check the solution. We use this local optimization if there are several columns sharing the same physical tree of routes. This technique can be used in addition to DP to generate columns for the master. We give the formulation of the local optimization below. By solving the local optimization model, we try to find the optimal start service time, traveling time along each edge and the loading quantity based on the physical routes of the current solution found by the master problem.

Sets:

- R_v — set of routes associate with ship v
- E_{sv} — set of edges included in route s for ship v

- N_{sv} — set of port visit included in route s for ship v

Parameters:

- X_{sv} — 1 if ship v takes route s, 0 otherwise. But this value can be fractional between 0 and 1.
- Y_{imk} — 1 if port visit (i, m, k) is visited by a ship, 0 otherwise
- B_k — upper bound of the time period which includes scenario tree node k
- $\underline{B}_k$ — lower bound of the time period which includes scenario tree node k
- K_v — capacity of ship v
- $\bar{SP}_{imkjnlsv}$ — sailing time for ship v at its lowest speed between (i, m, k) to (j, n, l)
- $\underline{SP}_{imkjnlsv}$ — sailing time for ship v at its highest speed between (i, m, k) to (j, n, l)

Here, X_{sv} does not need to be integer value. The value of X_{sv} is the same as the weight of schedule s in the solution of the master problem. It therefore can be a fractional value between 0 and 1.

Variables:

- $t^{sailing}_{imkjnlsv}$ — traveling time for ship s from port visit (i, m, k) to (j, n, l) along route s
- t_{imksv} — start service time for port visit (i, m, k) made by ship v in route s
- t^{S}_{imk} — start service time for port visit (i, m, k)
- q_{imksv} — loading quantity for port visit (i, m, k) made by ship v in route s
- q^{H}_{imk} — loading quantity for port visit (i, m, k)
- g_{imksv} — quantity on board ship v when the ship arrives port visit (i, m, k) in route s

Formulations:

$$\min \quad \sum_{v} \sum_{s \in R_v} \sum_{(i,m,k) \to (j,n,l) \in E_s} f_{ij}(t^{sailing}_{imkjnlsv}) \tag{6.5}$$

$$\sum_{v} \sum_{s \in R_v} X_{sv} t_{imksv} + (1 - Y_{imk}) B_k = t^S_{imk} \qquad \forall (i,m,k) \tag{6.6}$$

$$\sum_{v} \sum_{s \in R_v} X_{sv} q_{imksv} = q^H_{imk} \qquad \forall (i,m,k) \tag{6.7}$$

$$t_{imksv} + t^{sailing}_{imkjnlsv} - t_{jnlsv} \leq 0 \quad \forall v, s \in R_v, (i,m,k) \to (j,n,l) \in E_{sv} \tag{6.8}$$

$$\underline{SP}_{imkjnlsv} \leq t^{sailing}_{imkjnlsv} \; \forall v, s \in R_v, (i,m,k) \to (j,n,l) \in E_{sv} \tag{6.9}$$

$$t^{sailing}_{imkjnlsv} \leq \bar{SP}_{imkjnlsv} \; \forall v, s \in R_v, (i,m,k) \to (j,n,l) \in E_{sv} \tag{6.10}$$

$$\underline{B_k} \leq t_{imksv} \leq B_k \qquad \forall v, s \in R_v, (i,m,k) \in N_{sv} \tag{6.11}$$

$$g_{imksv} + q_{imksv} - g_{jnlsv} = 0 \; \forall v, s \in R_v, (i,m,k) - (j,n,l) \in E_{sv} \tag{6.12}$$

$$q_{imksv} \leq K_v \qquad \forall v, s \in R_v (i,m,k) \in N_{sv} \tag{6.13}$$

$$0 \leq g_{imksv} \leq K_v \qquad \forall v, s \in R_v, (i,m,k) \in N_{sv} \tag{6.14}$$

$$\textit{Inventory Constraints} \tag{6.15}$$

In objective function 6.5, f_{ij} is the cost function over traveling time for the ships. If f_{ij} is a strictly convex function as shown in formula 6.3, the local optimization problem is an NLP problem. However we can keep the problem linear by using a linear piece-wise approximation to f_{ij}. We calculate the start service time and loading quantity for each port visit in constraints 6.6 and 6.7. If a ship sails along a route, the start service time of a visit should be less than those of the later visits. This is shown in constraint 6.8. Constraints 6.9 and 6.10 give the lower and upper bound of the traveling time. And the bounds for the start service time at each port visits are given in constraint 6.11. Constraint 6.12 is the balance equation of amount of gas on board the ship. Constraint 6.13 means that the quantity of a single loading or unloading cannot exceed the capacity of the ship, and constraint 6.14 gives that the quantity on board a ship cannot exceed the capacity of the ship either. There are also a set of inventory constraints 6.15 involved in the model so as to keep the storage level within the lower and upper bound all the time. These inventory constraints are the same as the inventory constraints of the original model shown in Section 5.4. We do not repeat the formulations for these inventory constraints again here.

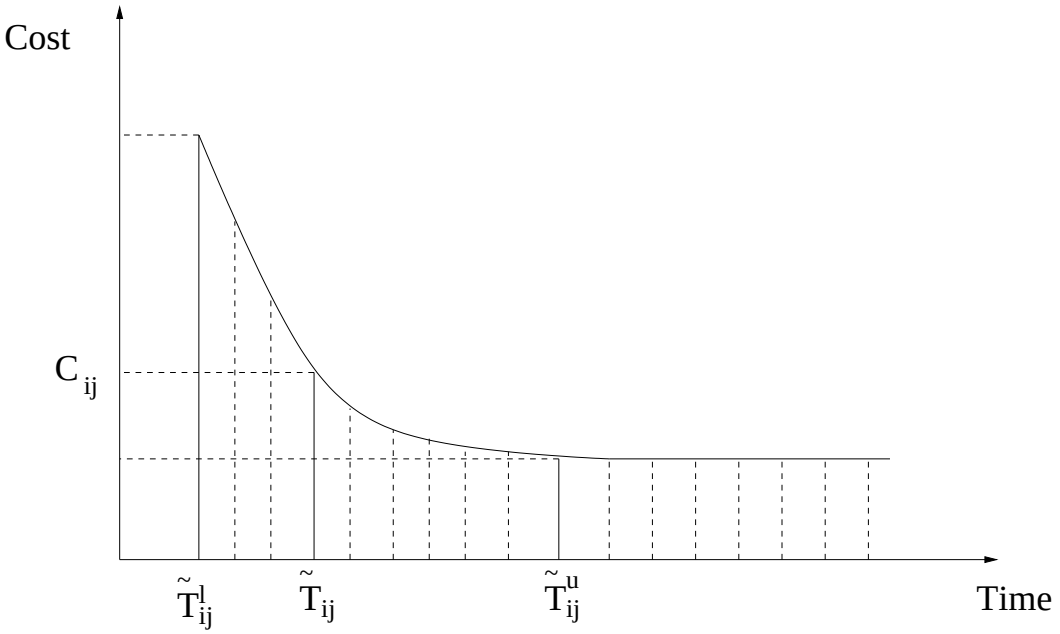


Figure 6.7: Example cost function

6.1.4 Computational Results

In order to test the variable speed model, some examples presented in Chapter 5 are modified and solved for the variable speed model. The only changes made to formulate the variable speed model for the examples in Chapter 5 are in subproblems. As discussed before, each traveling edge in the network is related to a decreasing nonlinear function.

In the variable speed model, ships can vary their speed and this can save traveling costs, since a situation where the fixed cost model sails then delays at the end of the journey, while the variable speed model can sail more slowly and complete the journey in the same time but at lower cost. Also the variable speed model allows the ship to travel faster than in the fixed speed model, which may be worthwhile because the ship can leave more time for the future journey and may get more savings later.

To build the cost function for the examples given in this section, we use the formula 6.3 to calculate the cost at each discrete time point. This is shown in Figure 6.7. Let the traveling time and cost in the original model be T_{ij} and C_{ij}, then we choose an arbitrary value of δ to get the traveling time with highest and lowest speed as following:

$$T_{ij}^{l} = T_{ij}(1 - 0.5\delta) \tag{6.16}$$

$$T_{ij}^{u} = T_{ij}(1 + \delta) \tag{6.17}$$

Then we discretise the traveling time and calculate the cost at each discrete time point as shown in Figure 6.7. These costs will be used when solving the DP in the ship subproblems.

Table 6.1 shows a comparison of computational results between the original and the variable speed model. In this case we take $\delta = 1$.

	Fixed Speed				Variable Speed			
	BB Node	Col	time	Obj	BB node	Col	time	Obj
b1	78	1251	13	1296.0	81	1363	18	1276.0
b2	177	3079	31	3300.0	345	4694	74	3089.3
b3	219	4204	47	2459.0	270	4494	70	2420.6
c1	81	2435	20	2699.2	151	3620	39	2531.2
c2	87	3948	26	2256.0	90	3882	35	2214.0
c3	237	4756	57	3408.0	251	6583	71	3037.0
d1	564	6206	120	3064.0	996	10255	225	2727.2
d2	63	1353	15	2185.3	108	1830	22	2090.4
d3	750	6945	138	2864.0	543	6029	118	2548.9
f1	405	9034	439	3022.0	411	8219	379	2928.9
f2	138	3623	126	3422.0	165	4790	192	2922.0
g1	342	7241	403	3145.5	195	5250	250	3099.7
g2	624	11557	705	2319.0	576	11350	683	2093.2
g3	132	4109	181	2854.5	198	4810	224	2723.7
h1	3598	30753	3690	6738.3	—	—	—	—
h2	2987	31983	3371	7020.0	—	—	—	—

Table 6.1: Comparison between original and variable speed model

In Table 6.1 we give the number of branch-and-bound nodes, number of columns generated, elapsed seconds and objective values for each example solved as the original and the variable speed model. From the results we can see that by using discrete traveling time in the cost function, the variable speed model does not seem to be more difficult to solve than the original model. We get lower objective values for the variable speed model as expected.

In Table 6.2 we give a comparison between the results with and without using local optimization. The only difference is the number of columns generated when solving the problem. The model with local optimization generates less number of columns in these examples.

	Without local-opt	With local-opt
	column	column
b1	1534	1363
b2	4871	4694
b3	4582	4494
c1	4200	3620
c2	3912	3882
c3	6778	6583
d1	12013	10255
d2	1913	1830
d3	6572	6029
f1	9855	8219
f2	4903	4790

Table 6.2: Comparison between results with and without local optimization

6.2 Divert model

6.2.1 Introduction

Another option for ships besides varying their speed is to change their destinations during sailing. In this thesis we call this the 'diverting case'. Since we may not know the future demand information before starting a journey, it may be advantageous to change the destination once the demand information becomes known. Therefore, developing a model based on stochastic optimization which can predict possible diverting cases and design the corresponding traveling schedules has the potential to reduce the traveling costs.

In the deterministic ship routing problem, the average of demands is calculated according to the historical data and used to solve the problem. For example, the scenario tree on the left hand side of Figure 6.8 could represent the possible demand situations, while the figure on the right hand side gives the average of these demands.

The solution of the deterministic problem with average demands might, for example, require a ship to sail from port C in period 1 to port A in period 2. However if the demand information is revealed at the beginning of period 2, and the demand in port B may be very high, then the ship may have to divert to visit port B to meet the demand. This situation is shown in Figure 6.9. We can see that the diverting case causes higher traveling costs than the non-diverting case. Because the future uncertainties are not considered by deterministic model, the expensive traveling cost caused by diversion cannot be avoided if a divert case happens.

In the stochastic ship routing problem, we deal with the problem based

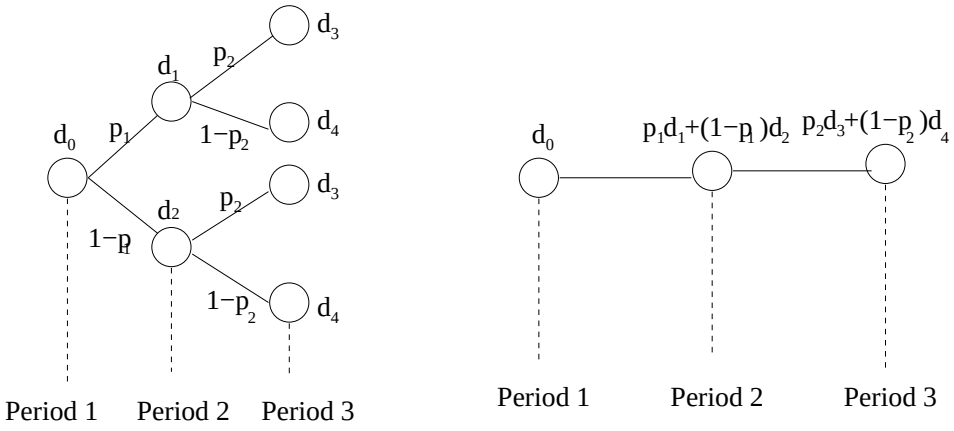


Figure 6.8: Scenario tree and average of demands

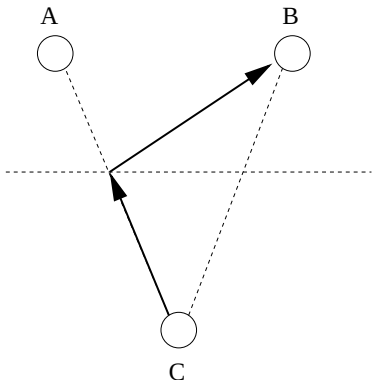


Figure 6.9: Divert situation

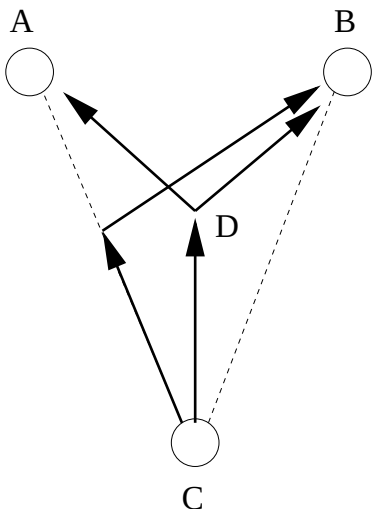


Figure 6.10: Divert: central points

on scenario tree by solving a deterministic equivalent of the stochastic model. This allows us to predict the possible diverting cases and find a suitable way to reduce the expected traveling costs. Once there is a possible diverting case, ships can first sail to some point at sea between the possible destinations at the end of the current period and decide which destination to go to at the beginning of the next period when the demand information for next period is known. See the Figure 6.10 as a continued example of Figure 6.9.

In the figure, the ship first sails to central point D, which is between ports A, B and C. Once the demand information is known, it sails from D to either port A or B. The total expected traveling cost of this is less than the cost of sailing towards A and sometimes having to divert to port B, which is the case shown in Figure 6.9.

In the original model of the ship routing problem we discussed before, when a ship finishes the service at a port in period T, there are 3 actions the ship can take: (1) it can sail to another port within the same time period and do another service there; (2) it can wait at the current port until the demand information for next period is revealed and decide the next destination to visit in the coming period; (3) it can leave the port immediately after the service and sail to another port but arrive at the next port in next period. When we allow ships to divert, the case (3) is replaced by the ship sailing toward a central point and the decision of the destination is left until the beginning of the next period. See Figure 6.11 and 6.12 for example of the non-divert and divert cases.

In the examples, there are two demand situations in period 2. In scenario node 2, demand at port A is high and the port needs to be visited shortly after the beginning of period 2, while in scenario node 3 no visit is needed to

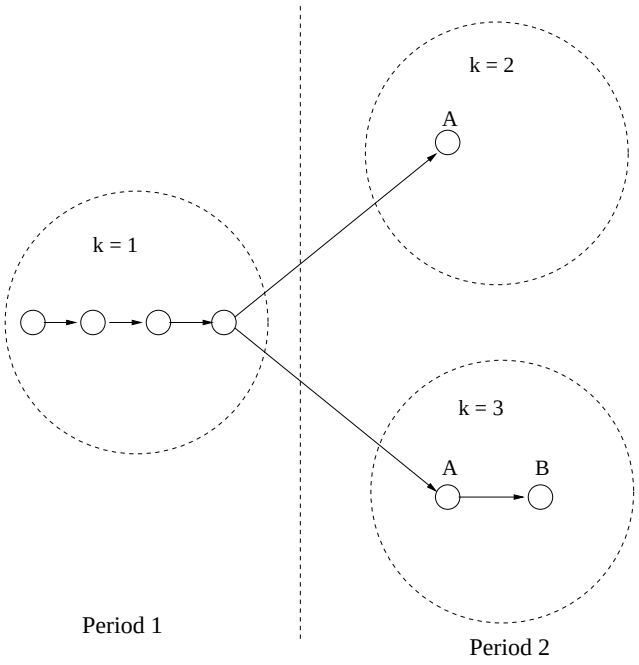


Figure 6.11: Non-divert case example

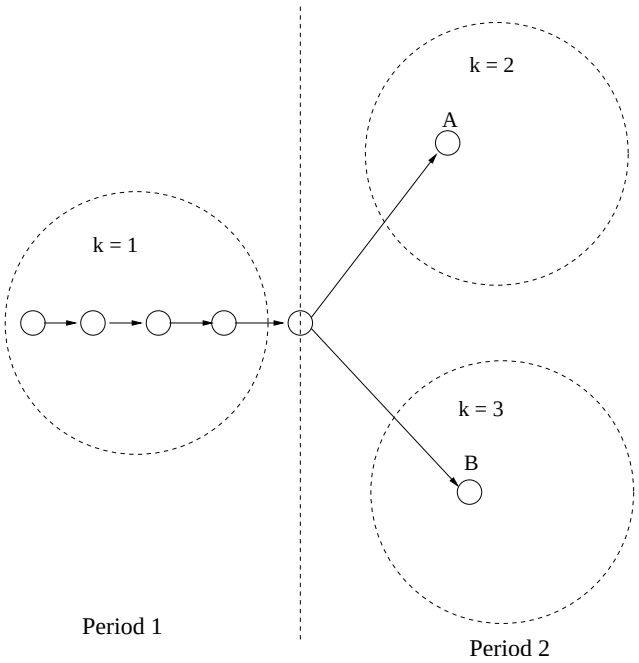


Figure 6.12: Divert case example

port A but port B needs a visit within period 2. Therefore, after the journey in period 1 a ship should sail to port A before the end boundary of period 1 so as to arrive at port A before the stock runs out. This is action 3 mentioned above. So in Figure 6.11, the ship has to sail to port A in both scenario node 2 and 3, and sail to port B to meet the demand after the visit to port A in scenario node 3. In this situation, extra traveling cost occurs because of the additional visit to port A in scenario 3. However, in the divert model ships are allowed to sail to central points in the sea when crossing the periods. So in Figure 6.12 the ship can sail to a central point among the ports until the future demand information is known. Then the ship can sail to port A in scenario node 2 and directly sail to port B in scenario node 3, which results in a cheaper expected traveling cost than the case given in Figure 6.11.

Among these possible actions, when crossing the period boundary action (2) has the best expected cost. It is cheaper than sailing to a central point and diverting to the destinations. For example, in Figure 6.10, a ship sets out from port C and has probability of P_1 to sail to port A in next period and has chance of P_2 to travel to port B. Suppose that the traveling cost is proportion to the distance. The expected cost is $P_1|AC| + P_2|BC|$ if the ship takes action (2), while it costs $P_1(|DC| + |AD|) + P_2(|DC| + |BD|)$ if it does not wait for the future information but chooses to sail to a central point D (Obviously we have $P_1(|DC| + |AD|) + P_2(|DC| + |BD|) \geq P_1|AC| + P_2|BC|$). However, when the expected cost of the whole route is considered the diverting case may be cheaper and in some case may be feasible when the non-diverting case is infeasible. For example, if the planning schedule is tight for a ship so that it has to visit some ports shortly after the beginning of next period and does not have enough time to wait for the future information in the current period. In this situation, the ship sails to a central point, which is a better choice than the case that sailing to one port and diverting to another according to the new coming information. This is the reason that we discuss the diverting model here. Note that action (3) above can be regarded as a particular case of sailing to a central point and diverting to the destinations and so does not have to be considered separately. In this case, the central point is the current port where the ship is in, so no sailing is involved.

In the real world, the forecast of the demand for product, such as oil and nature gas, is mainly in a regular basis, normally once a month. This provides us the possibility of building a stochastic model for the diverting case according to the prediction of the future demand. Because we use scenario tree to represent the future demand situations, our solution of the diverting model may be effected by the structure of the scenario tree. If a ship decides to leave the current port near the end of the current period, sails to another port and arrives at the port in next period, the chosen central point for the ship to divert may be very close to the current port, according to the solution of the model. In the realistic world, the ship may not choose the central points very close to the current port because doing that may not help much reducing

the expected traveling cost. But if the ship leaves the current port at the time that is not near the end of the period, then it will have more freedom to choose the central point for diverting and the chosen central point may be far away from the current port. In this case, there may be a possible saving of the expected traveling cost. Therefore, when we use the divert model introduced in this section, we may set a time interval (the upper bound of the interval is the period boundary) and do not allow ships to divert if the current time is within the time interval. Furthermore, if the forecast is not coming in a regular basis, we can rerun the diverting model once the new forecast arrives so as to let ships to take better schedules according to the future demand information.

The rest of the section is organized as follows: Section 6.2.2 presents the divert model based on the decomposition approach and solved by column generation method. And computational examples are shown in Section 6.2.3. Finally in Section 6.2.4 we introduce an alternative method for generating the central points a priori.

6.2.2 Divert Model with Decomposition Format

As discussed before, in the divert model instead of making its decision of which port to sail to next before the next period demand information is known, a ship can leave the current port immediately and sails to a central position in the sea until the future information is known. Then it can make the decision of the destination according to the known information. We introduce the divert model with the decomposition formulation in this section.

In order to add the diverting action into the decomposition formulation, we do some changes to the ship routing subproblems but nothing is changed in the master problem. In the original subproblems, ships can cross from one period to another by sailing to two types of sum-up nodes and then splitting into the corresponding demand scenario parts in the network. Sum-up nodes $\boxminus^{i-j}$ refer to the action that a ship leaves the current port, i, immediately after the service and sails to the destination port, j, in the future period before the future demand (action 3 mentioned in Section 6.2.1), while sum-up nodes $\boxdot^{i}$ refer to the action that the ship waits at the current served port, i, for the future demand information, and then sails to the destinations in the next period (action 2 mentioned in Section 6.2.1). In the divert model, there is another action, central point action, which means a ship sails to a central point immediately after the current service and diverts to the destinations when the demand information is revealed. Since central point action is better than action 3, a ship will no longer take the action 3 in the divert model. Because action 2 can be regarded as a particular case of the central point action, (in which the ship chooses the current port as the central point, sails to it with 0 traveling time and waits for the future information in the position) we can use a single $\square$ node for the central point action instead of sum-up node

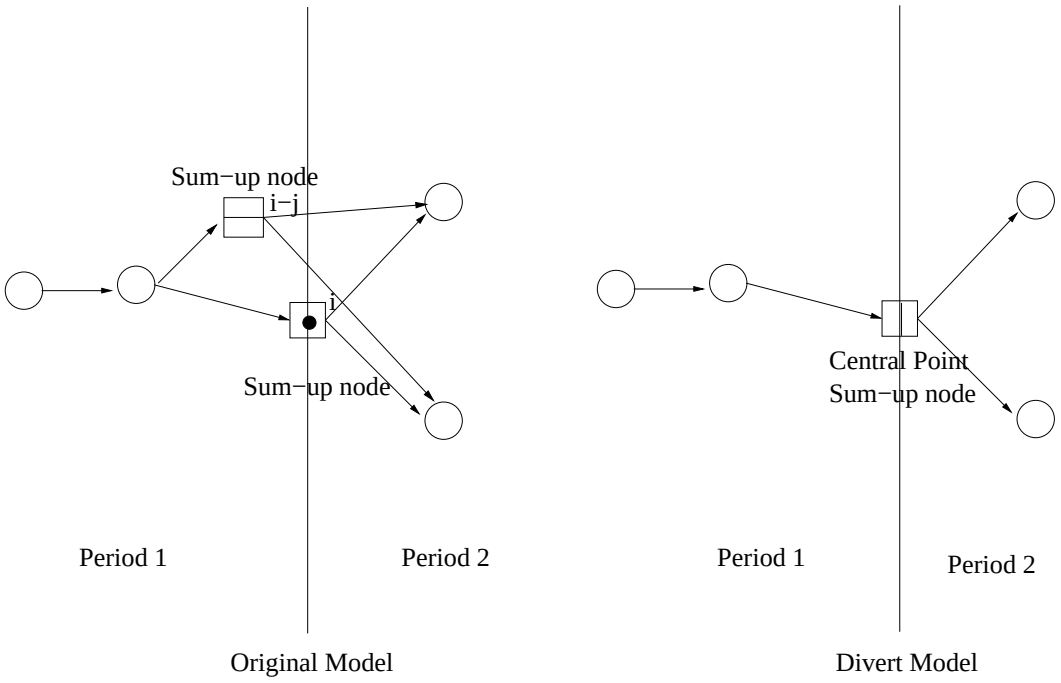


Figure 6.13: Comparison of DP networks

$\boxminus^{i-j}$ and $\boxdot^{i}$ in the network. This is illustrated in Figure 6.13.

A vertical line inside of $\square$ means the central point sum-up node is fixed on the period boundary but varied among different physical positions. Each central point sum-up node refers to a group of positions in the sea where the ship can arrive at the end of the current time period, and at which it gets the future information and diverts to the destinations.

Again choosing the suitable central positions for a ship to cross the time periods is important for the divert model and is not easy to do. It can increase the size of ship routing subproblem a lot and so increase the solving time for the subproblems. We cannot let ships choose any position in the sea as a central point for diverting, so we introduce a grid network to the DP and allow ships to choose suitable central points from a set of points on the grid. A grid network is shown in figure 6.14 as an example. In the example, we use the grid for the whole network space. The position in the network then can be represented by grid points. When a ship chooses the central points in DP, it can consider any possible grid point as candidate. The more detail grid network we use, the better the solution will be. However, there is always a trade-off between the quality of solution and problem size so that we have to choose a suitable grid network for the problem.

Because there are several destinations a ship can choose at a central point, the possible central points should be in between these destinations. For example, suppose that there are five destinations considered, only the

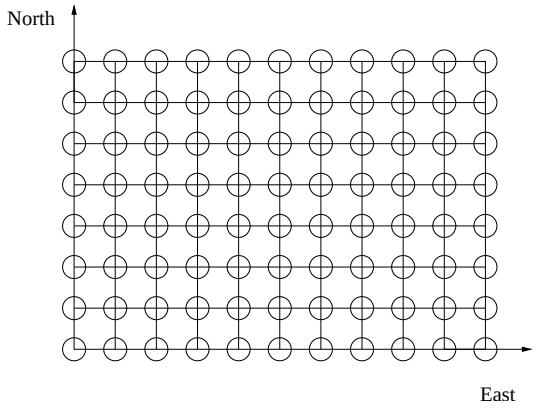


Figure 6.14: Grid network

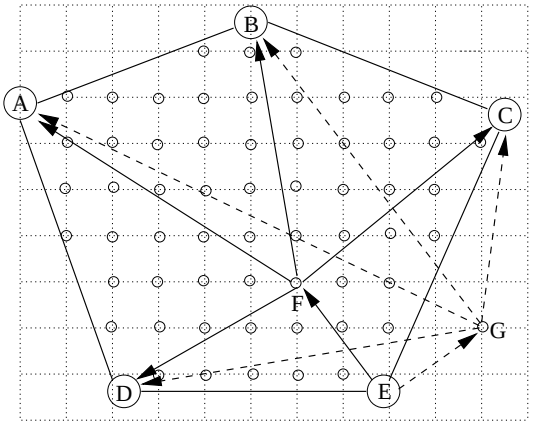


Figure 6.15: Possible central points in grid network

central points within the pentagon can be considered. This is shown in the Figure 6.15. It is clearly that sailing to a central point F within the pentagon $ABCDE$ is better than sailing to a point G outside the pentagon. Note that here we are assuming that there are no obstructions such as islands which would prevent a straight line journey between any two ports. If this is not the case a more general grid would be needed.

Associated with each possible central point in the grid network is a cost, which is the expected cost to the final dummy node. An example of cost function at the central point sum-up node in the divert model is given in figure 6.16.

The cost functions at the central point sum-up node and at the corresponding split nodes are the functions of the x and y coordinates of the point, but do not depend on time as time is fixed at the period boundary. The cost functions of other nodes except sum-up nodes and corresponding split nodes in the diverting model are the same as those in the original model.

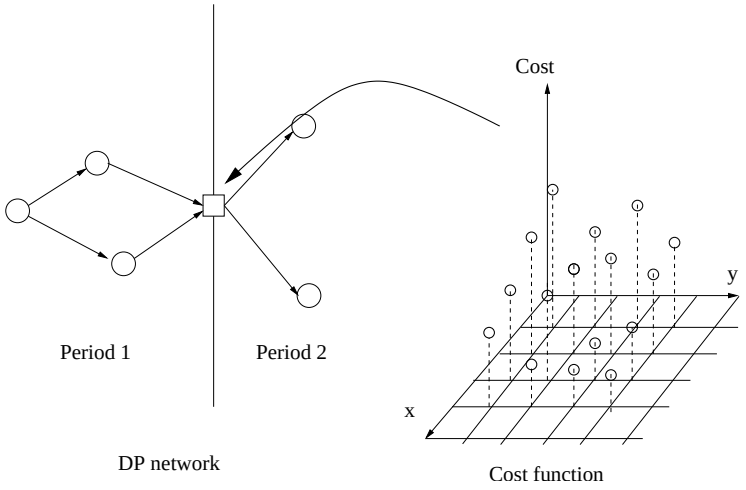


Figure 6.16: Cost function

When we update the cost function for the central point sum-up node from the corresponding split nodes in DP, the cost of a grid point is calculated by the sum of the cost at the same grid point on each split node linked with the sum-up node. This is shown in Figure 6.17.

The cost at a split node is updated by considering start service nodes linked with it. For each grid point at a split node, the cost is updated in the same way as the original model. An example is illustrated in Figure 6.18.

Figure 6.19 shows the way of updating node costs from central point sum-up nodes. For time T_1 not all the grid points at central point sum-up node can be used to update the cost, because some central points may be too far away from the end service node for the a ship to arrive at the point before the period boundary.

6.2.3 Examples and Discussion

In order to test the divert model using the decomposition formulation, several test problems were built here. A different grid network is used for each of these test problems. The information on these examples is listed in Table 6.3:

EX	Ports	Max Arrival	Scenario Nodes in tree	Planning Periods	Branches	Ships
B0-3	5	3	3	2	2	2
C0-3	6	4	7	3	2	3
D0-3	5	3	13	3	3	2

Table 6.3: Example Information

Here the examples with the same first letter are the same example but

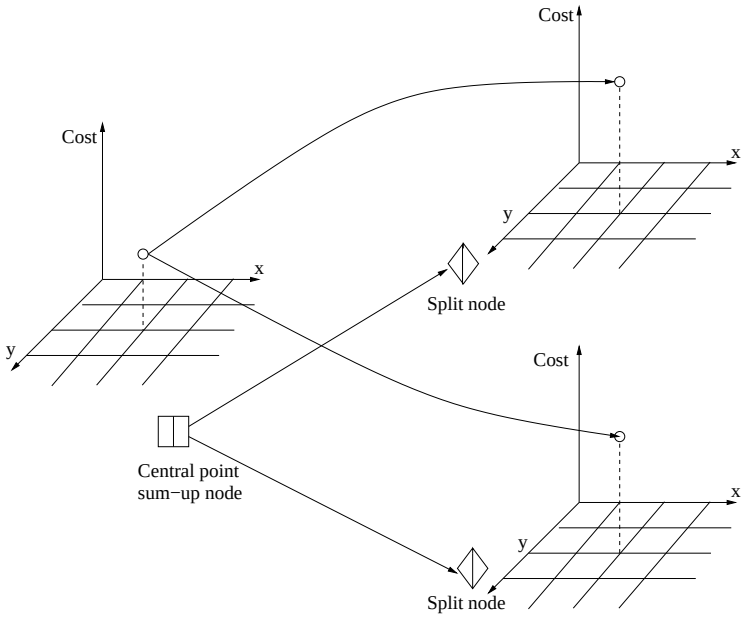


Figure 6.17: Update cost function for central point sum-up node

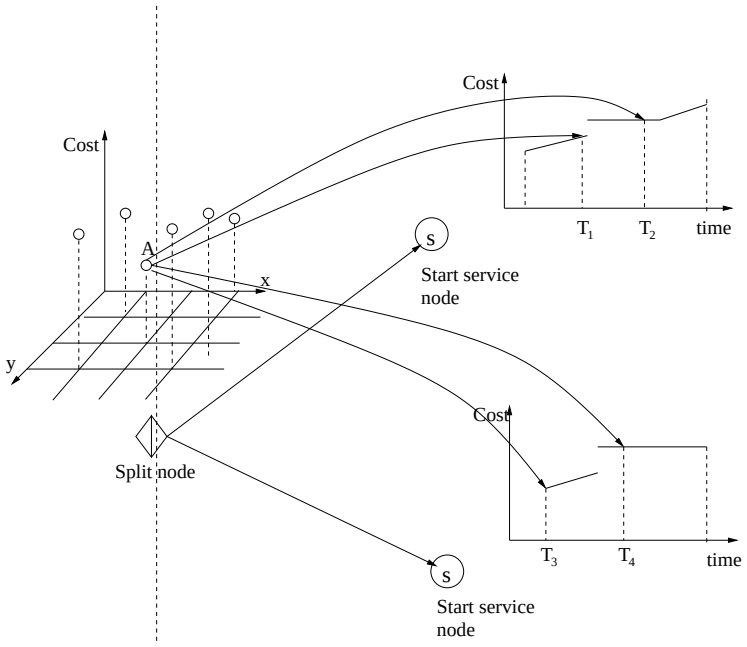


Figure 6.18: Update cost function for split node

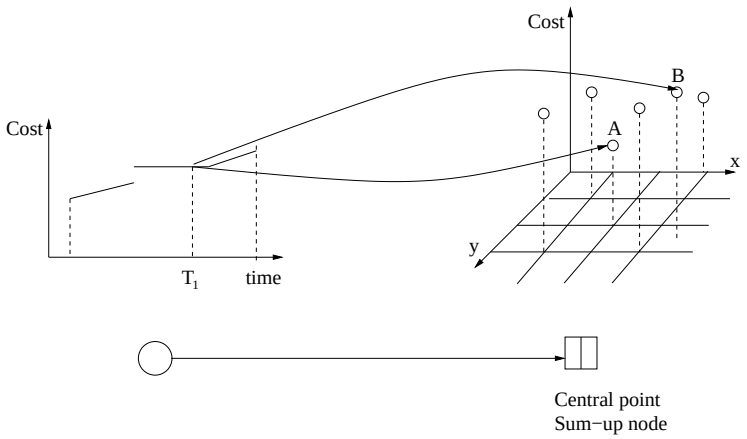


Figure 6.19: Update cost function for end service node from central point sum-up node

with different size of grid network used in subproblems. '*0' problems are fixed speed and no divertion models. In table 6.4, we show the computational results for these examples.

	BB Nodes	Columns	Seconds	Obj	central points
B0	72	1663	61	1721.2	–
B1	150	2818	111	1654.2	12
B2	186	3517	191	1542.8	16
B3	207	3693	262	1398.6	30
C0	817	8178	419	3580.0	–
C1	1356	10657	603	3520.0	12
C2	1608	13871	760	3520.0	19
C3	2145	19020	965	3350.0	37
D0	1321	11681	805	5950.0	–
D1	1705	15887	1002	5780.0	16
D2	1728	16088	1027	5620.6	25
D3	2546	22350	1305	5550.0	47

Table 6.4: Diverting model examples

'BB Nodes', 'Columns' and 'Seconds' show the number of branch-and-bound tree nodes used, number of columns generated for the master problem and the elapsed solving time respectively. 'central points' gives the number of central points considered in each example. The number of central points is determined by the size of grid network used. Here we use three different sizes of grid networks: the unit distances between two grid points are 1, 0.5 and 0.25 for problem 1, 2 and 3 respectively. From the computational results, we can

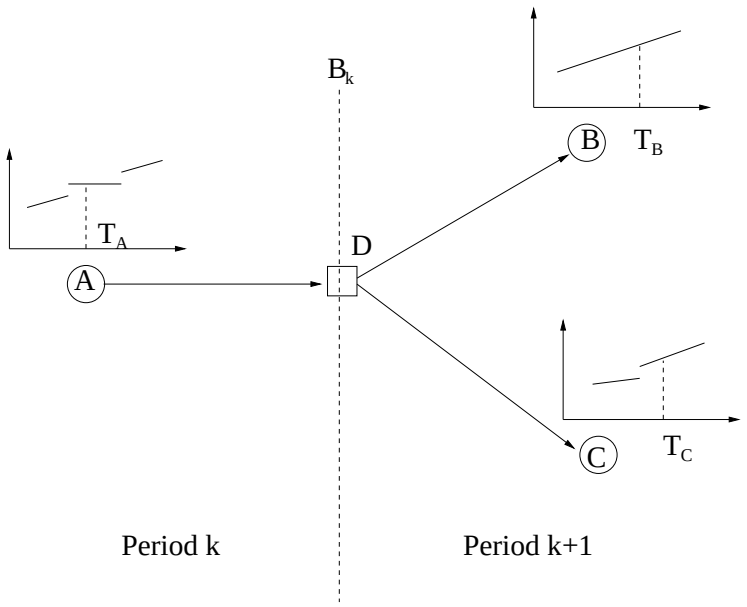


Figure 6.20:

see that a more detailed grid gives a better solution. From the table we can see that the non-diverting case is easier to solve, but gives worse objective values. The main reason for this is that in the non-diverting case, ships may sail to a port without demand information so that when the demand information is known, they have to visit additional ports to meet the demand requirements, which results in more expensive traveling costs.

6.2.4 Finding Central Positions with a Pre-Optimization Model

In the previous section, we discussed how to select central positions for the divert case by introducing a discrete grid network. This way allows us to keep the divert problems small enough to be solve, but we cannot ensure the optimality of solutions because the optimal place to be at a period boundary may not be one of the grid points. In this section, we introduce a method to generate optimal central points for different situations of crossing period boundaries by solving a group of nonlinear optimization problems a priori. We first introduce the nonlinear optimization model based on an example, and then the way of solving the subproblems is discussed.

Figure 6.20 is an example including two periods with three demand secnario parts. There are four nodes in the network, node A, B and C refer to ports and node D is the central point on the time period boundary.

Node A is in period k, and node B and C are in period $k+1$ but in different scenario tree nodes. P_{AB} and P_{AC} are the corresponding probabilities related

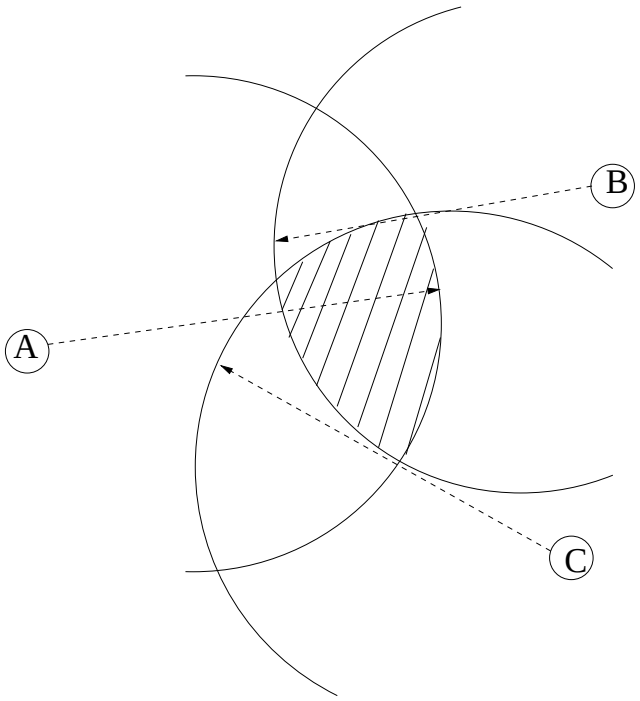


Figure 6.21: Possible central points area

to the scenario tree nodes. A ship leaves node A after the service, sails to a central point and diverts to node B or node C. Let T_A be the time the ship to be leaving port A, B_k the end of scenario node k and T_B and T_C the arrival times at port B and C. For each possible value of $B_k - T_A$, $T_B - B_k$ and $T_C - B_k$ the problem is to find the position in node D which minimizes the expected traveling cost.

As discussed, the sailing time between node A and D should be no more than $B_k - T_A$. But it does not mean that a ship has to use $B_k - T_A$ to sail, it can arrive node D at any time before B_k and wait for the new demand information unveiled. Similarly for edge (D, B) and (D, C), a ship can spend at most $T_B - B_k$ and $T_C - B_k$ on sailing along edge (D, B) and (D, C) respectively. Since ships can change their speeds during sailing in the divert model (suppose the highest speed is $\bar{V}$), they can arrive at any position at sea within a circle of radius $\bar{V}T$ within a time T. Back to the example, the biggest distance a ship can sail from node A, B or C is $\bar{V}(B_k - T_A)$, $\bar{V}(T_B - B_k)$ and $\bar{V}(T_C - B_k)$ respectively. Figure 6.21 illustrates the details. We can see that the possible central points should be in the shaded area, the intersection of the area within the circles.

The cost function of traveling was introduced in Section 6.1.1. We can use the cost function when we build the pre-optimization model here.

Now we can introduce the model which is used to minimize the traveling

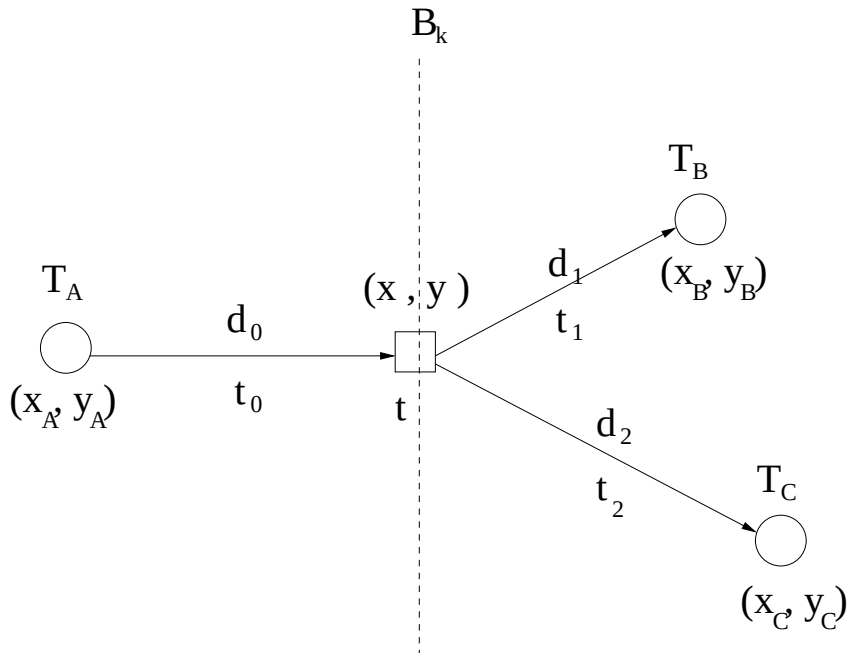


Figure 6.22: Example of finding central positions with pre-optimization

cost. We assume that $\bar{N}$ is the set of nodes which are not central points, in the network and $\bar{E}$ is the set of edges in the network. Let a be the index of a central point in the network. P_{ij} is the probability associated with edge (i, j). Let parameter (x_i, y_i) be the coordinates of port i, and let T_i be the start service time at node i. Finally, let $\underline{V}$ be the lowest speed for ships and $\bar{V}$ be the highest speed for sailing a unit distance.

There are several variables involved in the model. Let (x, y) be the coordinate of the central point, d_{ij} $((i, j) \in E)$ the length of edge (i, j) and t_{ij} the traveling time on edge (i, j). Then we assume the traveling cost on edge (i, j) is a function of distance d_{ij} and traveling time t_{ij}, $h_{ij}(d_{ij}, t_{ij})$. All these parameters and variables are shown in Figure 6.22 for an example.

In the example, $\bar{N} = \{A, B, C\}$ and $\bar{E} = \{(A, a), (a, B), (a, C)\}$. The formulation of the model for the example is:

$$\min \quad \sum_{(i,j)\in\bar{E}} P_{ij} h_{ij}(d_{ij}, t_{ij}) \tag{6.18}$$

$$s.t. \quad d_{Aa}^2 = (x - x_A)^2 + (y - y_A)^2 \tag{6.19}$$

$$d_{aB}^2 = (x_B - x)^2 + (y_B - y)^2 \tag{6.20}$$

$$d_{aC}^2 = (x_C - x)^2 + (y_C - y)^2 \tag{6.21}$$

$$0 \le t_{Aa} \le B_k - T_A \tag{6.22}$$

$$0 \le t_{aB} \le T_B - B_k \tag{6.23}$$

$$0 \le t_{aC} \le T_C - B_k \tag{6.24}$$

$$t_{ij}\underline{V} \le d_{ij} \le t_{ij}\bar{V} \quad (i,j) \in \bar{E} \tag{6.25}$$

Objective 6.18 minimize the total expected traveling cost. Constraints 6.19 – 6.21 calculate the distance d_{ij}. Constraints 6.22 – 6.23 ensure that for each node in the network, the arrival time should be no later than the start service time. The lower and upper bounds for the variables d_{ij} are calculated in Constraints 6.25. Figure 6.21 shows the situation of this constraint.

When solving a DP, each time we update node cost by crossing the boundary of a time period, model 6.18–6.25 is solved to find the best central point. Doing this requires us to solve a large number of nonlinear optimization models during solving DP. Instead before we start solve a DP, we can consider all the possible situations, including ports in different scenario nodes and start service time at each node and solve the nonlinear model. The optimal central points for all possible situations can be stored in a list. When updating node costs, we can just simply refer to the pre-solved results to get the best central points. So this solution method can fit well with our decomposition approach. Generating central points by this method requires extra time for solving a group of nonlinear optimization problems, but it can provide us the optimal central points for different situations of crossing the time period boundary. We do not implement this pre-optimization model in the thesis. The implementation of this model will be part of the future work.

6.3 Model with Bonus for Additional Work During Idle Time

In the original model we have introduced in Chapter 5, ships are allowed to wait outside ports without additional costs. Section 6.1 gives an extension model allowing ships to choose traveling speed during sailing. Because of the lower traveling cost with lower traveling speed, ships will reduce their speeds to save traveling cost instead of waiting outside ports. Another possible extension to the original model is to allow ships to use the idle waiting time to

do additional work (or tasks) to get extra income. This extension is introduced in this section. In real shipping industry, it is a good way to take advantage of waiting time to reduce costs or get additional reward. However, ships may not be able to do additional work during any idle time. So there are some limits for this activity:

- If the idle waiting time is short, it is impossible to do extra traveling to other ports.
- If there is one type of bulk commodity on board a bulk ship already, it is impossible for the ship to do the additional work transferring another type of bulk commodity because two different bulk commodities are not allowed to be mixed.
- For particular use ships, such as LNG tankers, it is not possible to do the addition work carrying other commodity even a ship is empty on board because the particular design of the ships is not suitable for these commodities.
- For bulk ships, it may cost a lot to clean up a ship before and after the additional work if the ship transfers a different bulk commodity. Therefore, ships may not do some work because of the expensive costs.

A ship can sail to a port, use waiting time to do additional work and return to the port to continue the scheduled travel. In the other case, a ship can also do additional work after the service to the current port, but it will sail to another destination port instead of going back to the port it was in before doing the work. There are two corresponding ways for ships to get the additional work information: (1) When a ship arrives a port, it can get local information about additional tasks. (2) Some information is known for ships anywhere, they can decide to take over some tasks anytime subject to the availability of these additional tasks.

In rest of the section we discuss possible ways to implement this based on our model. Several possible situations of the model including getting reward by doing extra work in idle time is listed below.

(1) The first situation is currently what we are doing in our original model. Giving a reward for completing their last service early encourages ships finish the services as early as possible. This situation could be easily extended to a model giving a reward at the end of every period. The changes need to implement this are only in the subproblems. However ships do not need to be empty to get this reward in the model, so this situation does not fit well with ships going to do the tasks with some commodity else.

(2) Giving a reward proportion to non-sailing time (with no lower bound of idle time) is the second possible way. The travel cost remains convex. Ships do not need to be empty to get the reward, and reward is the same for lots of small idle periods as for big idle periods of the same total time. This situation fits well with doing something else in the idle time. For implementation it is easy to be dealt with. No change is needed in the master problem. Without the lower bound of idle time, there will be no sudden jump-down in the cost function so that we do not need to do extra branching in the branch and bound algorithm.

(3) As the situation 2, we still give a reward proportion to idle time, but now with a threshold for idle time before a reward is given: i.e. there is a lower bound of idle time such that there is no reward given if the idle time is less than the lower limit. The reward is proportion to idle time if the non-sailing time is long enough. The threshold idle time makes the problem non-convex, there are jump-downs in the cost function. The master problem may choose a solution which is an average of two columns. And these two chosen columns have the same physical routes, but one column is without reward while the other has reward. So it may require more branching in the master problem to avoid this. It will encourage idle time to come in bigger chunks which may give more chances of finding alternative work.

(4) We can specify some non-inventory ports in the model, which are different from the ports with inventory bounds. These non-inventory ports can be visited between the inventory bounded ports. Then ships can buy and sell arbitrary amounts of commodity at non-inventory ports at given buying and selling prices and these ports can be visited any number of times. There are two possible situations: (a) Ships are empty at start and finish of visits to non-inventory ports. This gives the ship freedom to do any type of work and get reward. We only consider the time to finish the work when we try to decide which work to take. (b) A Ship has some amount on board at start and finish of visits to non-inventory ports. This requires the ship do the work transporting a certain amount of the same commodity as the one already on board between non-inventory ports. We need to consider both the time doing the work and the amount of commodity to transfer.

    Again changes only happen in the subproblems. However, costs could be non-convex because of the sudden jump-down in costs, so it will require more branching to ensure that different physical routes containing non-inventory ports are not mixed. Mixing the same physical routes in the master problem is valid so does not require branching.

(5) This situation is the most difficult situation, in which we consider the time windows on non-inventory ports. There is a number of cargo

transfers of buying and selling opportunities within time windows. Because of the time windows, repeats of the same opportunities are not possible, so they have to be monitored in the master problem with extra constraints. If the extra opportunities are available throughout a period (this means that we set the time windows for these opportunities to the whole period) but with an overall upper bound on the total amount. Then they can be dealt with by adding inventory constraints in the master problem for each non-inventory port or each transport leg. Non-inventory port cost would be similar to the inventory port case but there would be no flow into or out of the port in the period except that ships visit them and transfer amount of commodity.

All these five situations discussed above give possible ways to deal with the case of extra reward for additional work during idle time in our decomposition model. Implementation of these situations is not included in this thesis but will be for future work.

Chapter 7

Summary and Conclusions

In this thesis, we have discussed a stochastic ship routing problem with inventory management. This problem involves a fleet of ships transferring a single commodity from a set of supplier ports to a set of consumer ports. The demands at consumer ports are uncertain, and the objective of the problem is to find the schedule for each ship so that the consumer demands are satisfied and the total traveling cost is minimized. Different solution approaches for the problem have been studied in the thesis. The computational results show that the problem is a complex combinatorial problem and difficult to solve, however by using a column generation method medium sized problems can be solved to proven optimality.

The initial approach we took to the problem formulated as a mixed integer programming model. In the model, two binary variables were used to control the ship status in the traveling control constraints, one for ship positions and the other for ship sailing information. A group of continuous variables in the inventory constraints are used to monitor the stock level in the port inventory as well as on board each ship. The MIP model was written in AMPL and solved directly by calling the commercial optimization software package CPLEX 10.0. Formulating the problem by MIP model is quite straightforward and very easy to understand and implement. However, our computational results show that it is very time-consuming to solve the problem using this MIP formulation even for some small sized examples.

In order to solve bigger problems within a reasonable time, we have proposed a Branch and Price algorithm. A master problem is formulated as a set partitioning model with additional inventory constraints, while subproblems, one for each ship, are solved by stochastic dynamic programming to find the least reduced cost columns for the master problem. The optimal discrete solution is found using Branch and Bound, and a column generation method is used to solve the relaxed LP iteratively in each Branch and Bound node.

As far as we know this is the first time the stochastic ship routing problem with uncertain demands and inventory management has been studied. A

similar deterministic problem has been studied before and our decomposition approach when applied to deterministic problem is similar to that, the major difference being that we include the inventory constraints in the master problem instead of dealing with them in a separate subproblem. We do this because that the relaxed LP can be solved very efficiently by CPLEX and even including the inventory constraints, the time for solving the master problem can still be negligible. In the deterministic problem the optimal solution is a schedule for each ship and there is a column in the master problem for each ship schedule that has been generated. In our stochastic formulation the optimal solution is a tree of schedules for each ship and there is a column in the master problem for each tree of schedules that has been generated.

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.

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.

To capture more realistic situations in our problem, we have introduced several extensions to the original model. A variable speed model allows ships to use different speed for different legs of the journey and to change their traveling speeds during a single leg of the journey. This flexibility gives a fuel cost saving because the fuel cost depends significantly on the speed. To implement the variable speed case, changes have be made to the ship subproblems. Local optimization was used to find the solution with optimal timing and quantities loaded. Compared to the original model, lower objective costs are found using the variable speed model. The main reason for this is that ships can slow down to save fuel instead of sailing in high speed and waiting outside ports in the original model. Our current version of the variable

speed model can only get a sub-optimal solution. This is because only a fixed speed is allowed for the trips that cross the period boundaries. However, this problem can be resolved by the divert model which we introduced.

In the divert model ships can change their destinations and speed during sailing. This is advantageous on the leg of the journey where new demand information becomes available during the leg. In this thesis we have given examples to show why diverting case occurs. Three different possible solving approaches are discussed. An MIP model was easy to formulate. This involves first calculated the possible diverting points and added them into the model as dummy ports without demand. Adding these dummy ports makes the model very difficult to solve. However a much better approach is to extend the decomposition formulation. We introduced a new type of sum-up node in the DP network and used a discrete grid of points as possible central points for diverting. Then we build a new form of cost function for these new sum-up nodes based on the discrete grid network. Another possible way based on the decomposition formulation was to find central positions with pre-optimization model. A group of nonlinear optimization problems can be solved to find optimal diverting position a priori by considering all the possible journey times when crossing a period boundary.

Another extension model has been introduced in the thesis, but not implemented. This is a model that gives a bonus for a ship doing additional contract work during idle times of its schedule.

In future work, we will study a set of possible methods to speed up the column generation process. Generating columns for the master problem heuristically is one possible way to speed up the solution. This may be specially useful for problems which are too large to solve to proven optimality. If an IP model is built for generating the least reduced cost columns for the master problem, it can avoid cycles in the solutions automatically. However, when we tried this, the IPs took a very long time to solve. The methods which let us solve the IP models more efficiently are worth studying in the future.

As for the diverting model, methods will be explored to speed up the solution. Computational experiments will be done for the model with bonus for additional work during idle time. One further possible area for future work based on the models introduced in this thesis is to integrate these models together so that we can capture larger and more realistic situations in the shipping industry.

Bibliography

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows*. Prentice-Hall.

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.

Azaron, A. and Kianfar, F. (2003). Dynamic shortest path in stochastic dynamic networks: Ship routing problem. *European Journal of Operational Research*, 144:138–156.

Baker, B. and Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30:787–800.

Baker, E. (1983). An exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–945.

Balakrishnan, N. (1993). Simple heuristics for the vehicle routing problem with soft time windows. *Journal of Operational Research Society*, 44(3):279–287.

Bausch, D., Brown, G., and Ronen, D. (1998). Scheduling short-term marine transport of bulk products. *Maritime Policy and Management*, 25:335–348.

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.

Bent, R. and Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

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

Bertsimas, D. and Simchi-Levi, D. (1996). A new generation of vehicle routing research: Robust algorithms, addressing uncertainty. *Operations Research*, 44(2):286–304.

Bramel, J. and Simche-LEVI, D. (1995). A location based heuristic for general routing problems. *Operations Research*, 43:649–660.

Bramel, J. and Simchi-Levi, D. (1997). On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Operations Research*, 45:295–301.

Bronmo, G., Christiansen, M., and Nygreen, B. (2006). Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society advance online publication*.

Calvo, R. W. (2000). A new heuristic for the traveling salesman problem with time windows. *Transportation Science*, 34(1):113–124.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers and Operations Research*, 33:2972–2990.

Chiang, W.-C. and Russell, R. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63:3–27.

Cho, S.-C. and Perakis, A. (2001). An improved formulation for bulk cargo ship scheduling with a single loading port. *Maritime Policy and Management*, 28:339–345.

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. (1996). Inventory and time constrained ship routing. *PhD Thesis NTNU*.

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.

Christofields, N., Mingozzi, A., and Sandi, C. (1979). *Combinatorial Optimization*. John Wiley & Sons, Ltd.

Christofields, N., Mingozzi, A., and Toth, P. (1981a). An algorithm for the time constrained traveling salesman problem. Report IC-OR-81/12, Imperial College of Science and Technology, London.

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

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

Dantzig, G. and Wolfe, P. (1960). The decomposition algorithm for linear programming. *Operations Research*, 8:101–111.

Desaulniers, G., Lavigne, J., and Soumis, F. (1998). Multi-depot vehicle scheduling with time windows and waiting costs. *European Journal of Operational Research*, 111:479–494.

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.

Dumas, Y., Desrosiers, J., Gelinas, E., and Solomon, M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371.

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

Fagerholt, K. (2001). Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, 131:559–571.

Fagerholt, K. (2004). A computer-based decision support system for vessel fleet scheduling – experience and future research. *Decision Support Systems*, 37:35–47.

Fagerholt, K. and Christiansen, M. (2000a). A combined ship scheduling and allocation problem. *Journal of the Operational Research Society*, 51(7):834–842.

Fagerholt, K. and Christiansen, M. (2000b). A travelling salesman problem with allocation, time window and precedence constraints – an application to ship scheduling. *International Transactions in Operational Research*, 7:231–244.

Fisher, M. (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642.

Fisher, M., Jörnsten, K., and Madsen, O. (1997). Vehicle routing with time windows – two optimization algorithms. *Operations Research*, 45:488–492.

Gelinas, S., Desrochers, M., Desrosiers, J., and Solomon, M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109.

Gendreau, M., Hertz, A., Laporte, G., and Stan, M. (1998). A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, 43(3):330–335.

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.

Gendreau, M., Laporte, G., and Seguin, R. (1996). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477.

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.

Guo, Z. and Mak, K. (2004). A heuristic algorithm for the stochastic vehicle routing problems with soft time windows. *IEEE*, pages 1449–1456.

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

Ioachim, I., Gélinas, S., Soumis, F., and J.Desrosiers (1994). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *GERAD and École Polytechnique, Montréal, Canada, H3T1V6 23 pp.*

Irnich, S. and Desaulniers, G. (2004). Shortest path problems with resource constraints. *Les Cahiers du GERAD.*

Irnich, S. and Villeneuve, D. (2006). The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing.*

Kall, P. and Wallace, S. (1994). *Stochastic Programming.* JOHN WILEY & SONS.

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

Kohl, N. and Madsen, O. (1997). An optimization algorithm for the vehicle routing problem with time windows based on lagrangean relaxation. *Operations Research*, 45:395–406.

Langevin, A., Desrochers, M., Desrochers, J., and Soumis, F. (1993). A two commodity flow formulation for the traveling salesman and makespan problems with time windows. *Networks*, 23:631–640.

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

Mingozzi, A., Bianco, L., and Ricciardelli, S. (1997). Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations Research*, 45(3):365–377.

Osman, I. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451.

Pesant, G., Gendreau, M., Potvin, J., and Rousseau, J. (1996). An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Publication CRT-96-15, Centre de recherche sur les transports, Universite de Montreal, Montreal.*

Potvin, J.-Y. and Bengio, S. (1996). The vehicle routing problem with time windows – part ii: Genetic search. *INFORMS Journal on Computing*, 8(2):165–172.

Potvin, J.-Y., Kervahut, T., Garcia, B., and Rousseau, J.-M. (1996). The vehicle routing problem with time windows – part i: Tabu search. *INFORMS Journal on Computing*, 8:158–164.

Potvin, J.-Y. and Rousseau, J.-M. (1995). An exchange heuristic for routing problems with time windows. *Journal of Operational Research Society*, 46:1433–1446.

Ronen, D. (1986). Short-term scheduling of vessels for shipping bulk or semi-bulk commodities originating in a single area. *Operations Research*, 34:164–173.

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

Russell, R. (1995). Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Science*, 29(2):156–166.

Savelsbergh, M. (1985). Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305.

Semet, F. and Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, 41:469–488.

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.

Solomon, M. (1987). Algorithm for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM Scocity for Industrial and Applied Mathematics.

Xie, X., Wang, T., and Chen, D. (2000). A dynamic model and algorithm for fleet planning. *Maritime Policy and Management*, 27:53–63.