# NETWORK DESIGN AND ALLIANCE FORMATION FOR LINER SHIPPING

A Thesis
Presented to
The Academic Faculty

by

Richa Agarwal

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Algorithms, Combinatorics, and Optimization

School of Industrial and Systems Engineering
Georgia Institute of Technology
August 2007

# NETWORK DESIGN AND ALLIANCE FORMATION FOR LINER SHIPPING

Approved by:


Professor Özlem Ergun,
Committee Chair
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Professor George L. Nemhauser
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Professor Özlem Ergun, Advisor
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Professor Michael D. Meyer
School of Civil and Environmental Engineering
*Georgia Institute of Technology*

Professor Ellis Johnson
School of Industrial and Systems Engineering
*Georgia Institute of Technology*

Professor H. Venkateswaran
College of Computing
*Georgia Institute of Technology*



Date Approved: August 2007

*To my parents,*

# ACKNOWLEDGEMENTS

I would like to express my sincerest thanks and gratitude to my advisor, Dr. Özlem Ergun. Without her continuous guidance, encouragement and financial support this work would not have been possible. She provided me not only academic advisement and guidance but has also been a true mentor, friend and inspiration. To Dr. Ergun, I am forever grateful.

I would like to thank Dr. Ellis Johnson, Dr. George L. Nemhauser, Dr. Michael D. Meyer and Dr. H. Venkateswaran for serving on my dissertation committee and providing valuable feedback. I am quite fortunate to have had the opportunity to present my work before and receive advice from such esteemed academicians.

I would like to thank current and former fellow students (Amandeep, Burak, Divya, Gultekin, Jin, Kong and Yi) at Georgia Tech for frequent discussions and for their friendship. Special thanks to two truly amazing friends, Kapil Gupta and Dylan Shepardson, for their continuous support and encouragement. I must also express my heart-felt appreciation for my friends (not in ISyE), who made my days in Atlanta truly memorable and enjoyable.

I thank my parents, my sisters Ruchi and Charu and my brother Ayush for their unconditional love and support in all the decisions I have made. Most importantly I would like to express my sincere gratitude to my husband, Himanshu; it is with him by my side that life is more enjoyable and everything seems easy. There is no way I could have done this without him.

There are countless others, at Georgia Tech and otherwise, who have contributed to this thesis in various ways. Thank you.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

In maritime transportation, liner shipping accounts for over 60% of the value of goods shipped. However, very limited literature is available on the study of various problems in liner shipping. In this thesis we focus on problems related to this industry.

Given a set of cargo to be transported, a set of ports and a set of ships, a common problem faced by carriers in liner shipping is the design of their service network. We develop an integrated model to design service network for the ships and to route the available cargo, simultaneously. The proposed model incorporates many relevant constraints, such as the weekly frequency constraint on the operated routes, and emerging trends, such as obtaining benefits from transshipping cargo on two or more service routes, that appear in practice but have not been considered previously in literature. Also, we design exact and heuristic algorithms to solve the integer program efficiently. The proposed algorithms integrate the ship scheduling problem, a tactical planning level decision, and the cargo routing problem, an operational planning level decision, and provide good overall solution strategy. Computational experiments indicate that larger problem instances, as compared to the literature, can be solved using these algorithms in acceptable computational time.

Alliance formation is very common among global liner carriers however a quantitative study of liner alliances is missing from literature. We provide a mathematical framework for the quantitative study of these alliances. For the formation of a sustainable alliance, carriers need to agree on an overall service network and resolve issues concerning distribution of benefits and costs among the members of the alliance. We develop mechanisms to design a collaborative service network and to manage the interaction among the carriers through the allocation of profits in a fair way. The

mechanism utilizes inverse optimization techniques to obtain resource exchange costs in the network. These costs provide side payments to the members, on top of the revenue generated by them in the collaborative solution, to motivate them to act in the best interest of the alliance while satisfying their own self interests.

# CHAPTER I

# INTRODUCTION

*Sea cargo* is the freight carried by ships. It includes anything traveling by sea other than mail, persons and personal baggage. Sea cargo transportation is a cheap, safe and clean transportation mode, compared to other modes of transportation. Rates for sea cargo transportation are approximately one-tenth of air freight rates. Increasing globalization and inter-dependence of various world economies is leading to a tremendous positive growth in the sea cargo industry. International and domestic trade of many nations depends on this mode of transportation and it has helped many nations such as Singapore to shape their export-dominated economy. According to [5], in United States, which is the largest trading nation in the world for both imports and exports - accounting for nearly 20% of world trade, sea cargo is responsible for moving over 99% of the international cargo. U.S. ports and waterways handle more than 2.5 billion tons of trade annually, and that volume is projected to double within the next fifteen years.

As depicted in Figure 1, the increase in sea-borne trade worldwide has led to similar trends in the growth of the world fleet. Although the fleet mix and size have changed over time considerably, for example the recent use of bigger ships, the main motive of the industry remains the same - efficient utilization of the ships. The increasing sea-borne trade and the increasing size of the world fleet presents new challenges for planners and demands attention from researchers to develop optimization based decision support systems for efficient fleet management and routing of the cargo.

Next, we discuss some of the emerging trends in this industry. Prior to that, a

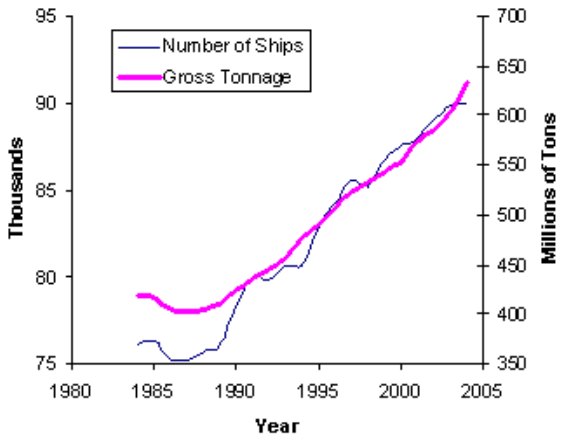


**Figure 1:** Growth of the world fleet and sea-borne trade. Source: Lloyd's Register of Shipping

couple of definitions are in place. A sea *carrier* is a person, business or organization that offers transportation services via sea on a worldwide basis. A *shipper* is a person or company who is either the supplier or the owner of the cargo that is to be shipped. Today sea carriers offer a global transport solution to a global shipper with global shipping needs.

## *1.1 Emerging Trends in Sea Cargo*

The sea cargo industry is going through many changes that are reshaping the face of this industry forever.

- One of the most prominent changes is the increasing use of containers. *Containerized* cargo is the cargo that have been physically and economically stowed in a container. Containerization of cargo minimizes port labor, maximizes ship's capacity utilization and has revolutionized the sea cargo industry. The dimensions of containers have been standardized and the term twenty foot equivalent unit (TEU) is used to refer to one twenty foot long container. For example a 20 feet long container is expressed by 1 TEU and a 60 feet long container is expressed by 3 TEUs. According to [24], at the start of 1980s only around 20%

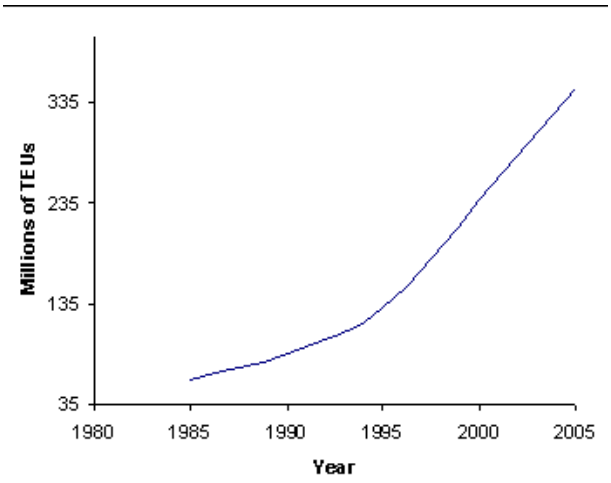


**Figure 2:** Growth in container movements worldwide. Source: Drewry Container Market Quarterly

of all general cargo shipments were carried in containers. In 2001 this figure increased to 60%. An IBM white paper [45] shows that container shipping market is still growing at 8-10% per year. This trend is also depicted by Figure 2.

A serious challenge associated with container shipping is the repositioning of empty containers. Because of the huge imbalance in demand on some trade routes, carriers need to reposition empty containers. For example, there is a high demand of commodities in America from Asia but not vice versa thus, carriers need to reposition empty containers from America to Asia. This process involves huge costs. According to [51], a 10% reduction in equipment and repositioning costs can potentially increase profitability by 30-50%.

- Sea cargo industry has seen a tremendous growth in number as well as in size of *transshipment ports*. A transshipment port is a port where cargo is transferred from one ship to another which is bound for different local, regional or global ports. Through a sequence of moves by cranes, cargo is transferred either directly from a ship to another or temporarily stored at the port before being loaded onto outbound ships for further transportation. Use of containers

makes this transfer very convenient and cost effective. Transshipment services provide carriers with additional routing options, reduced transit times and act as facilitator of international trade. For example the Hutchinson terminal in Freeport, Bahamas has become a major transshipment port between the Eastern Gulf Coasts of the United States, the Gulf of Mexico, the Caribbean, South America, and trade lanes to European, Mediterranean, far Eastern and Australian destinations. Other major international transshipment ports include Singapore, Port Klang in Malaysia and Hong Kong. In 2003, approximately 30% of worldwide container movements were transshipped and it is believed that this number is rising. According to [65], 80% of all containers handled at the port of Singapore, world's second largest container port and world's busiest port in terms of shipping tonnage, are transshipments.

- Traditionally, companies have focused on their own resources and ability to perform effectively and efficiently. However recently the many and varied carriers in the sea-cargo industry, who in the past worked independently of each other, are working in close liaison. Shipping alliances allow carriers to realize economies of scale, extend customer base and increase asset utilization while providing customers with more frequent sailings. Since 1990, when Sea-Land and Maersk introduced the alliance system and began sharing vessels in the Atlantic and Pacific oceans, mergers have become increasingly common. Recently, smaller alliances are collaborating to form even bigger alliances, for example The Grand Alliance and The New World Alliance laid down foundations for cooperating in 2006.

## *1.2 Liner Shipping*

Global shipping can be distinguished into three different modes of operation - *industrial*, *tramp* and *liner*. In industrial shipping, the shipper owns the ships and aims to

minimize the shipping cost. In tramp shipping, a carrier engages in contracts with shippers to carry bulk cargo between specified ports within a specific time frame. Additional cargo (if any available in the market) are picked depending on the fleet capacity to maximize revenue. In liner shipping, a carrier decides on a set of voyages, makes the schedule available to shippers and operates on it. Thus, one can identify industrial shipping with "owning a car," tramp shipping with "a taxi service" and liner shipping with "a bus service" with definite schedules and a published itinerary.

As the focus of this thesis is liner shipping, we now discuss it in detail. Liner shipping mainly involves carrying containerized cargo on regularly scheduled service routes. Liner services involve higher fixed costs and administrative overhead than for example tramp shipping because tramps usually wait until they are full before departing from a port whereas liner services promise to depart on predetermined schedules regardless of whether the ship is full. The number of ships required for a given liner service route is determined principally by the frequency required on the service route, the distance travelled by a ship on the route and the speed of the ship. For example, a weekly liner service between New York and Hamburg may require four ships to maintain the necessary frequency.

As observed by [15], liner shipping is growing at a high pace with the increasing global container traffic. In the United States, in 2003, liner shipping with its network of ships, containers, port terminals and information systems, handled over 60% of the total sea-borne trade. According to [8] between January 2000 and January 2006, the TEU capacity deployed on global liner trades has risen from 5,150,000 TEUs to 9,135,000 TEUs, a 77.4% increase.

Liner shipping involves decision making at strategic, tactical and operational planning levels. Figure 3 outlines the key decisions that need to be made at different levels of the planning horizon.

The first stage of planning is to determine an optimal mix and size of ships in

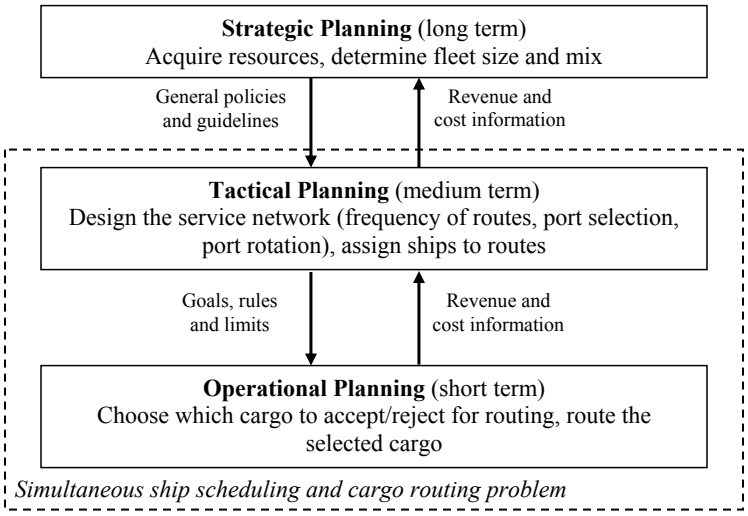


**Figure 3:** Different planning levels for liner shipping

the fleet. Given that a ship involves huge capital investment (usually millions of US dollars) and the cost of idling a 2,000TEU ship is $20,000-$25,000 per day, the strategic problem is extremely important.

The tactical planning decision involves decisions regarding an optimal design of service routes, i.e. the sequence of ports to be serviced by the given fleet and the assignment of ships to these routes. Ships move in *cycles* from one port to another following the same port rotation for the entire planning horizon. We refer to this problem as *the ship scheduling problem*.

The operational planning problem involves decisions regarding which cargo to accept or reject and which path(s) to use to ship the selected cargo. We refer to this problem as *the cargo routing problem*. A carrier may select not to transport some cargo, either because it is not profitable or because there is other cargo, perhaps at other ports, that is relatively more profitable. A cargo starts its trip from an in-land location and may or may not visit an intermediate port before arriving at its *origin port*. This network which utilizes trucks or railroad to bring cargo from an inland location to ports or ships to bring cargo from intermediate ports to origin ports is known as the *feeder network*. Cargo then moves from its origin port to its *destination*

*port*, possibly after visiting some intermediate ports. From there it is taken to its final in-land destination using another feeder network. Some of the intermediate ports that a cargo visits during its journey from the origin port to the destination port may act as transshipment ports where transfer of cargo from one ship to another takes place.

The decisions made at one planning level affect decision making at other planning levels as well. The decisions at the strategic level set the general policies and guidelines for the decision making at the tactical and operational levels. Similarly, the decisions at the tactical level set the limitations and network structure for the operational planning level. In the reverse direction, the information (for example cost and revenue) provide the much needed feedback for decision making at a higher level. Given a fleet size and mix, the service network laid at the tactical planning level governs which routes can be formed at the operational planning level to route cargo. The cargo picked at the operational planning level and the routes selected determine the cost and revenue that can be generated and thus the profitability of the given service network. These two problems are highly inter-dependent and thus it is important that they be studied in an integrated framework.

Collaboration and alliance formation is a common phenomenon among liner shipping operators. Carriers used *conferences*, as a means for curbing competition and controlling tariff rates in the market, as early as 1875. More recently, carriers are forming strategic alliances that allow them to realize economies of scale, extend their customer base and increase asset utilization while providing customers with more frequent sailings and faster transit times ([64]). Alliances account for more than half of the liner services on major global routes. In the mid 1990s an estimated 60% of the total global liner capacity was accounted by alliances. Typically members of an alliance pool their ships on a particular trading route and allocate part of each ship's capacity to the alliance members. As a result of these alliances and agreements, shipments arranged through one carrier may actually be moved by a ship operated

by another and alliance members can offer higher sailing frequencies than would be possible using only their own ships. Alliances are most common on deep sea routes such as the Asia-North America route that require a bigger commitment in terms of assets (ships) from carriers. Also, alliances are used to achieve cost efficiencies, especially in terminal operations.

Alliances lead to many large scale problems at the tactical and the operational planning level such as managing a large pool of ships, as contributed by the members of the alliance, and designing a large scale network to satisfy multiple demands due to various carriers. Members of an alliance together decide on a set of routes to operate. Also they need to decide how to realize these routes i.e. how should the different members of the alliance assign their ships to the selected routes. Further they need to decide how to share the benefits of the alliance. Thus, successful alliance formation requires allocation algorithms for distributing benefits, costs, and assets' capacity among the members of the alliance in a fair way to motivate them to "play along."

## *1.3 Contributions of the Thesis*

The contribution of this thesis is two fold. First, we present an integrated study of the tactical and the operational planning level problems in liner shipping. As shown in Figure 3 we refer to this problem as *the simultaneous ship scheduling and the cargo routing problem*. Our goal is to account for the emerging trends in liner shipping (containerization and transshipments) in our model and to solve the large scale optimization problem efficiently. Second, a detailed study of the alliance formation among liner carriers is performed.

In the first part of the thesis (Chapter 2) we present an integrated model, a mixed integer linear program, to schedule ships in a given fleet and efficiently route the maximum possible cargo, simultaneously. Our model handles many relevant constraints

and emerging trends that appear in practice but have not been considered previously. For instance, customers expect carriers to provide them a regular schedule by maintaining at least weekly frequency on the routes they operate on. This constraint however has not been considered in the literature. Use of containers to ship cargo has revolutionized the international cargo shipping industry. However, huge imbalance in demand in various world economies leads to the problem of empty container repositioning. The proposed model ensures weekly frequency on the operated routes and has the flexibility to incorporate empty container repositioning. Our model also allows cargo routes to encompass a combination of service routes rather than a single service route, thus providing carriers with increased routing opportunities. We are not aware of any earlier results on transshipment of containers at an intermediate port from one ship to another. We consider a heterogenous fleet with ships of different sizes, cost structures and speeds. As is common in container shipping, we allow multiple pickup and delivery on ships. These features allow the use of our model in a wide variety of problem settings.

The proposed integer program is too large to be solved economically by general mixed integer programming codes, thus we develop algorithms to solve it efficiently. More specifically, a greedy heuristic, a column generation based algorithm and a two phase Bender's decomposition based algorithm is developed and their computational efficiency in terms of the solution quality and the computational time taken is discussed. These algorithms exploit the separability of the problem and utilize the fact that the ship scheduling problem can be reduced to a cycle generation problem and the cargo routing problem can be reduced to a multicommodity flow problem. We also propose an efficient iterative search algorithm to generate schedules for ships. Computational experiments are performed on randomly generated instances simulating real life with up to 20 ports and 100 ships and the results are presented. Our computational results are encouraging and establish that the algorithms developed

here can be used to solve larger instances, as compared to the literature.

The second part of the thesis concerns with the challenges offered by alliances. As carriers form alliances by pooling their ships and integrating their networks, the maximum overall revenue that an alliance can generate can be obtained by replacing individual carriers with one large carrier, with a fleet equal to the combined fleet of the individual carriers and a demand structure equal to the combined demand of all carriers. We refer to this problem as *the optimization problem for the grand coalition* and to its solution as *the collaborative optimal solution*. Though working in collaboration, carriers cannot be assumed to follow the collaborative optimal solution but their own self-interests. For example on collaborative routes the "resource," e.g. capacity on a ship, belongs to some carrier who does not allow other carriers to freely obtain "benefits" from using it. Thus, for sustainable alliances, the task is not only to solve large scale optimization problems but also to provide algorithms to share the benefits of alliances in such a way that the best overall solution is obtained and all carriers are motivated to collaborate. The split of income and costs is an intriguing research topic and very little is available in literature on the systematic study of alliances among sea carriers. We provide algorithms to guide the members of an alliance in deciding which routes to operate, how to operate the chosen routes, how to distribute the capacity on ships and how to share the benefits from the alliance.

Chapter 3 presents the motivation behind alliance formation among carriers in liner shipping. We then study a special case of the problem and utilize the linear programming machinery to allocate the overall benefit generated by the alliance among its members in a "fair" way. The notion of "core," which guarantees that the sum of benefits allocated among the members is their maximum attainable benefit and no subset of members can collude and obtain better benefits for its members, is utilized to define fairness.

Study of collaboration among carriers in liner shipping led us to the study of a more

general *multi-commodity flow game*. In a maximum multi-commodity flow problem given a set of demand to be satisfied and capacity on the edges of the network the objective is to maximize the revenue generated by simultaneously satisfying different demands in such a way that the total amount of flow through each edge is no more than the edge's capacity. In the multi-commodity flow game, demand in the network and the capacity on the edges is owned by different members. The goal of each member is to use the underlying capacity in the network to satisfy his own demand. However, members often need to exchange capacity in the network to realize this goal. This framework where a number of participants interact in a multi-commodity flow setting occurs in many real life applications in transportation and logistics. In Chapter 4, we present a mechanism that facilitates capacity exchange on the edges of the network among the alliance members in a multi-commodity flow game setting. Specifically, the mechanism provides capacity exchange costs on the edges of the network to motivate a member who owns the capacity on an edge to sell it to another member who can utilize that capacity to satisfy his own demand. In a special case, when each of the network edges is owned by an unique member we show that the allocation made by our mechanism provides an allocation in the core.

Finally, we present membership mechanisms that allocate resources and benefits to carriers for forming sustainable alliances in liner shipping. In literature, although some qualitative studies are available, a quantitative study of liner shipping alliances is missing. In Chapter 5, we provide a mathematical framework to study alliance formation among liner carriers. We compute an optimal collaborative solution for the alliance by solving the network design problem for a large fictitious carrier with demand and fleet equal to the sum of the demand and fleet of individual carriers. However, individual carriers in the alliance cannot be assumed to accept the optimal collaborative solution. More over, for an individual carrier, the revenue generated by satisfying demand and the cost incurred on the collaborative routes is often not

enough to motivate him to behave in the best interest of the alliance. We develop algorithms that provide side payments to carriers so that the best overall revenue is generated for the alliance. This is achieved by finding capacity exchange costs (as in Chapter 4) so that a carrier who has unused space on his ships is motivated to sell it to carrier who can use it to transport his cargo. Our computational results suggest that the mechanism can be used to help carriers form sustainable alliances.

# CHAPTER II

# SHIP SCHEDULING AND NETWORK DESIGN FOR CARGO ROUTING IN LINER SHIPPING

## *2.1 Introduction*

*Sea cargo* is the freight carried by ships. Increasing globalization and inter-dependence of various world economies is leading to a positive growth in the sea cargo industry. Today, sea carriers offer transportation services on a worldwide basis. Increasing sea-borne trade (a 3.8% annual growth in 2005) and the increasing size of world fleet (a 7.2% increase in deadweight tons in 2005) offer new challenges and problems for the planners. Deadweight tons measures the weight of the ship at any loaded condition minus the weight of the ship with no crew, cargo, fuel etc on board.

Among the different modes of shipping, liner shipping is on a continuous rise. Liner services carry over 60% of the value of goods shipped by sea [23]. In liner shipping a carrier decides on a set of voyages, makes the schedule available to shippers and operates on it. As mentioned in Chapter 1, liner shipping mainly involves carrying containerized cargo, i.e. the cargo that have been physically and economically stowed in a container. Containerization of cargo makes the handling of cargo easy, minimizes port labor and maximizes ship's capacity utilization. Also, cargo is allowed to travel on more than one service route before reaching its final destination. A port where cargo is transferred from one ship to another for further transportation is referred to as a transshipment port. The number and size of transshipment ports is on a continuous rise. Further collaboration among liner shipping carriers is very common. Today smaller alliances are collaborating to form even bigger alliances, for example The Grand Alliance and The New World Alliance laid down the foundation for cooperating

in 2006. The trend of consolidating fleets and service routes demands better decision support systems to control a large fleet of ships and to solve large scale scheduling and optimization problems.

Liner services involve higher fixed costs and administrative overhead than for example tramp shipping because tramps usually wait until they are full before departing from a port whereas liner services promise to depart on predetermined schedules regardless of whether the ship is full. The number of ships required for a given liner service route is determined principally by the frequency required on the service route, the distance travelled by a ship on the route and the speed of the ship. For example, a weekly liner service between New York and Hamburg may require four ships to maintain the necessary frequency.

As mentioned in Chapter 1, liner shipping involves decision making at strategic, tactical and operational planning levels. In the strategic planning stage, the optimal number and mix of ships in a fleet is determined. Given that owning a ship involves a huge capital investment (usually in millions of US dollars) and the cost of idling a 2,000TEU ship is \$20,000-\$25,000 per day, the strategic level decisions are extremely important. In this chapter, we study the tactical and the operational level decisions.

In the tactical planning stage, the service network is designed by creating the ship routes, i.e. the sequence of port visits by a given fleet, and the assignment of ships to these routes. Ships move in *cycles* from one port to another following the same port rotation for the entire planning horizon. To maintain a customer base and to provide customers with a regular schedule most carriers have at-least one departure each week from each port visited on a service route (i.e. a cycle). This requires that the number of ships that operate on a cycle be at-least equal to the number of weeks that it takes to complete the cycle. Some cycles such as those connecting Asia to North America, may take up to eight weeks to complete, which means that a carrier requires at-least eight ships to introduce a new service on such a route. The problem of designing the

service network of a carrier is referred to as the *ship scheduling problem*.

In the operational planning stage, a carrier makes decisions regarding which cargo to accept or reject for servicing and which path(s) to use to ship the selected cargo. This is referred to as *the cargo routing problem*. A carrier may elect not to transport some cargo, either because it is not profitable or because there is other cargo, perhaps at other ports, that is relatively more profitable. A cargo starts its trip from an in-land location and arrives at its *origin port*. This network which utilizes trucks, railroads, or waterways to bring cargo from an inland location to its origin port is known as the *feeder network*. Cargo then moves from its origin port to its *destination port*, possibly after visiting some intermediate ports. From there it is taken to its final in-land destination using another feeder network. Some of the intermediate ports that a cargo visits during its journey from the origin port to the destination port may act as transshipment ports where cargo is transferred from one ship to another.

The decisions made at one planning level affect the decision making at other planning levels as well. The decisions at the strategic level set the general policies and guidelines for the decision making at the tactical and operational levels. Similarly, the decisions at the tactical level set the capacity limitations and network structure for the operational planning level. In the reverse direction, the information on cost and revenue that are generated by the system given the set parameters provides the much needed feedback for decision making at a higher level.

Over the years, the sea cargo industry has been conservative in terms of adopting new decision support systems. It has a long tradition of manual planning by experienced planners. More over, in general ship scheduling involves a large variety of problems. Hence, mostly tailor made models for specific problems with specialized constraints and objectives are available in the literature. Furthermore, most of the available literature have been developed for industrial and tramp shipping ([15]). Because of the many differences in modelling and problem structure itself, it is difficult

to draw comparisons between the existing literature. We next briefly overview a set of representative papers related to container and liner shipping. For a comprehensive review of literature on ship scheduling and cargo routing we recommend [52] for the work done before 1983, [53] for the decade 1982-1992 and [15] for the last decade.

[50] provides a nonlinear integer program to maximize total profit by finding an optimal sequence of ports to visit for each container-ship and an optimal number of cargo units to be transported between each pair of ports by each ship. They allow multiple pick ups and delivery on their ships. However, a special network structure with a restriction on loading and unloading of cargo at the end ports is considered. Furthermore, the model does not consider transshipments by not allowing cargo to be carried on ships that do not visit either the port of origin or the port of destination. They report that their algorithms solve instances with 3 ships and up to 20 ports within an hour.

[27] considers the liner shipping problem in a special network where all cargo is transported from a set of production ports to a single depot. The problem is solved by first generating all feasible single ship routes, and then solving a set partitioning problem. Again the model does not allow for transshipments. Although a weekly frequency constraint is imposed on the operated routes, the feasible routes for the particular problem considered have a maximum route time of one week only. Thus on any of the feasible routes a single ship can maintain weekly frequency. Instances with up to 19 ships on a network with up to 40 ports are reported to be solved within a couple of seconds.

Finally, [49] provides a review of linear and integer programming models, that only consider the deployment of a fleet of liner ships, with different ship types, on a set of predetermined routes with targeted service frequencies to minimize operating and lay-up costs.

As noted earlier, decisions made at one planning level affect decision making

at other planning levels. Given a fleet size and mix, the service network laid at the tactical planning level governs which routes can be formed at the operational planning level to route cargo. The cargo picked at the operational planning level and the routes selected determine the cost and revenue that can be generated and thus the profitability of the given service network. These two problems are highly interdependent and thus it is important that they be studied in an integrated framework. In this chapter, we present a new mixed integer programming (MIP) model for the integrated ship scheduling and the cargo routing problem for containerized cargo. We refer to this problem as *the simultaneous ship scheduling and the cargo routing problem*. Since the proposed integer program is too large to be solved economically by general mixed integer programming codes, we develop algorithms to solve it efficiently.

Our model handles many relevant constraints and emerging trends that appear in practice, but have not been considered previously in the literature. For instance, customers expect carriers to provide them a regular schedule by maintaining at least a weekly frequency on the routes they operate on. To the best of our knowledge, this constraint has not been considered in the literature in its full generality. We successfully impose the weekly frequency constraint at the ports visited by a carrier. As is common in container shipping, we allow multiple pickup and delivery on our ships i.e. we allow containers loaded at one port to have more than one port of destination. More over, the fleet of a carrier usually consists of various ship types with different characteristics that may change over time. We consider a heterogeneous fleet with ships of different sizes, cost structures and speeds. In the literature, although there are references (see for example [27]) to models with a heterogeneous fleet, most of these models consider ships with identical service speeds. Repositioning of empty containers efficiently is a big problem in liner shipping. Our model with some modifications has the flexibility to incorporate empty container repositioning. Empty container repositioning has been studied by [62] and [14] also, however these

papers consider only the movement of empty containers on a given network. We also allow for cargo routes to encompass a combination of cycles rather than a single cycle (with some simplifying assumptions on the cost of transshipment), thus providing carriers with increased routing opportunities. We are not aware of any earlier results on transshipment of containers at an intermediate port from one ship to another. These features allow the use of our model in a wide variety of problem settings.

In the most general approach for solving ship scheduling problems to date, [27] generates a set of feasible schedules by including non-linear and intricate constraints, and then solves a set partitioning problem. Our goal in this chapter is to model the simultaneous ship scheduling and cargo routing problem in its generality and solve it for large-scale instances. Hence, rather than being limited to an initial set of routes or exhaustively listing all the routes for ships, we design algorithms that exploit the separability of the problem to iteratively generate good cycles for ships and efficiently route the demand. More specifically, we utilize the fact that the ship scheduling problem can be reduced to a cycle generation problem and the cargo routing problem can be reduced to a multicommodity flow problem. We develop a greedy heuristic, a column generation based algorithm and a two phase Benders decomposition based algorithm and compare the computational effectiveness and efficiency of these approaches.

Our computational results are encouraging and establish that some of the algorithms developed can be used to solve larger instances, as compared to the literature, in terms of fleet size that arise as a result of collaborations and mergers in the sea cargo industry. We report computational results on problem instances with up to 100 ships and 20 ports.

The rest of the chapter is organized as follows. The next section introduces our notation, mathematical formulation and a note on the complexity of the problem. Three different algorithms, a greedy heuristic, a column generation based algorithm

and a Benders decomposition based algorithm, are discussed in Section 2.3. Section 2.4 provides various algorithmic and implementation details. Computational experiments are presented in Section 2.5. Conclusions and directions for future work are discussed in the final section.

## 2.2 Problem Description

We now present a mathematical formulation for the simultaneous ship scheduling and containerized cargo routing problem after introducing our notation and a space-time network.

Let $\mathcal{P}$ denote the set of ports. We will treat demand as a set of commodities with a positive supply at the origin ports and a positive demand at the destination ports. Each such commodity is characterized by an origin port, $o$, a destination port, $d$, the day of the week, $i$, when the supply is available at port $o$, the maximum demand (in TEUs) that may arise at port $d$, $D^{(o,d,i)}$, and the revenue obtained by satisfying one TEU of the demand, $R^{(o,d,i)}$. We use the triplet $(o, d, i)$ to identify a particular demand and we let $\Theta$ be the set of all such triplets. We call such triplets *demand triplets*.

A carrier typically has several different types of ships in his fleet. Each *ship type* usually has different capacity and speed and specifies the characteristics of a group of ships that are considered identical. We denote by $\mathcal{A}$ the set of all the ship types and use the index $a$ to represent a particular ship type. We associate the following information with each ship type $a \in \mathcal{A}$: $T^a$ denotes the capacity of a ship in TEUs for a ship of type $a$, for $p, q \in \mathcal{P}$, $l^a_{(p,q)}$ denotes the number of days it takes for a ship of type $a$ to make a sailing from port $p$ to port $q$ and $N^a$ denotes the number of ships of type $a$ available in the given fleet.

Given that the temporal aspects of the problem are important, we formulate the simultaneous ship scheduling and cargo routing problem as a multicommodity flow

problem with side constraints on a *space-time* network. Furthermore, we use days as our time units in the space-time network, since in general the transoceanic routes do not visit more than one port in a given day. Let $G = (V, E)$ be a directed space-time network with vertex set $V$ and edge set $E$. Each vertex $v \in V$ represents a port, $port(v)$, on a day of the week, $time(v)$. That is, for each port $p \in \mathcal{P}$ we create seven vertices in $V$. For notational convenience, we associate a subscript with each vertex, i.e. $v = v_{(p,i)}$ where $port(v) = p$ and $time(v) = i$. We refer to the vertices of $G$ either by $v$ or $v_{(p,i)}$ depending on the ease of exposition.

The network $G = (V, E)$ contains three types of edges. The first is the set of *ground edges*. For every ship type $a \in \mathcal{A}$, we construct ground edges by connecting nodes $v_{(p,i)}$ to $v_{(p,i+1)}$ $\forall p \in \mathcal{P}$ and $1 \leq i \leq 6$. We also connect $v_{(p,7)}$ to $v_{(p,1)}$ $\forall p \in \mathcal{P}$. For a ship, these edges represent an over-night stay at a port and for cargo they represent an overnight stay at a port either on ground, or on the same or on a different ship before continuing further. Next, for every ship type $a \in \mathcal{A}$ and pair of ports $p, q \in \mathcal{P}$, we construct *voyage edges*, $(v_{(p,i)}, u_{(q,j)})^a$ for $1 \leq i, j \leq 7$ such that $i - j = l^a_{(p,q)}\ mod(7)$. The voyage edges represent the movement of ships and cargo from one port to another at a given speed. Finally, we create a set of *fictitious edges*, $(v_{(d,j)}, u_{(o,i)})$ for all demand triplets $(o, d, i) \in \Theta$ and $1 \leq j \leq 7$. An edge $(v_{(d,j)}, u_{(o,i)})$ only allows the flow of commodity $(o, d, i)$ on it and enables us to view the flow of commodity $(o, d, i)$ in the network as a circulation. Let us denote the set of all ground edges by $E_g$, the set of all ground edges for ship type $a$ by $E^a_g$, the set of all voyage edges by $E_v$, the set of all voyage edges for ship type $a$ by $E^a_v$ and the set of all fictitious edges by $E_f$. That is, $E_g = \bigcup_{a \in \mathcal{A}} E^a_g$, $E_v = \bigcup_{a \in \mathcal{A}} E^a_v$ and $E = E_g \cup E_v \cup E_f$. We also use the following additional notation: $InEdges(v)$ denotes the set of incoming edges into vertex $v$ and $OutEdges(v)$ denotes the set of outgoing edges from vertex $v$; for an edge $e = (u, v) \in E$, $tail(e)$ denotes vertex $u$ and $head(e)$ denotes vertex $v$. Figure 4 represents a space-time network with four ports and two cycles, $C_1$ and $C_2$. Note

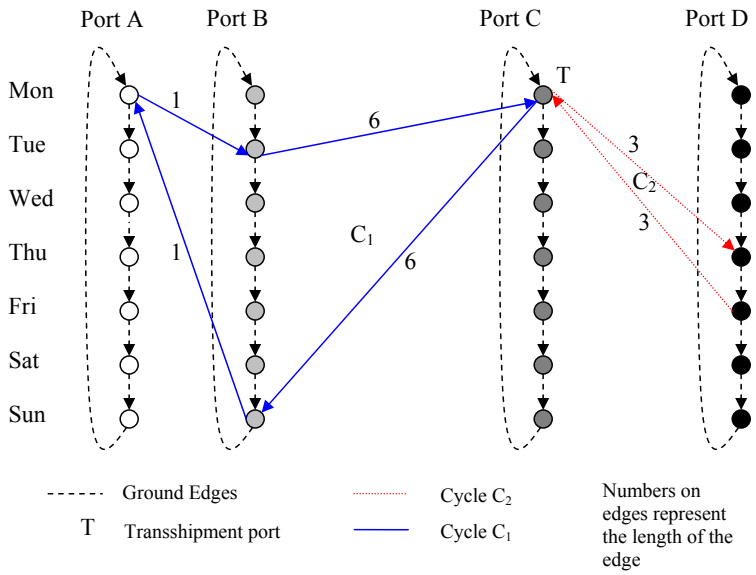


**Figure 4:** Network with four ports and two cycles

that port C acts as a transshipment port to transport cargo from port A to port D.

The length of voyage edge $e = (v, u) \in E_v^a$, $l_e^a$, is equal to the number of days it takes for a ship of type $a$ to reach from $port(v)$ to $port(u)$. We also let $l_e = 1$ for $e \in E_g$ and $l_e = 0$ for $e \in E_f$. The capacity of an edge represents the total amount of flow in TEUs that the edge can sustain. Ground edges at a port may have finite or infinite capacity depending on whether we wish to impose a limit on the amount of cargo that can be handled/stored at a port. Capacity on a voyage edge depends on the number of ships (and their capacities) that cover the edge.

There are various fixed and variable costs associated with the simultaneous ship scheduling and cargo routing problem. While some of these costs are incurred by ships, others are incurred by cargo. For the costs related to ports, we let $c_v^{s,a}$ be the one-time cost incurred by a ship of type $a \in \mathcal{A}$ when visiting $port(v)$ and $c_v^c$ be the total cost that a TEU of cargo incurs at $port(v)$ per day. The port visit costs may be different at different ports. Similarly, $c_e^{s,a}$ reflects the cost of operating a ship of type $a$ on edge $e$ in deep sea if $e \in E_v^a$ and the cost of an overnight stay for a ship of type $a$ at $port(head(e))$ if $e \in E_g^a$. For cargo, $c_e^c$ for $e \in E_v$ reflects the cost of shipping a

TEU of cargo on edge $e$ and for $e \in E_g$ reflects the cost of storing or holding a TEU of cargo at $port((head(e))$. Fictitious edges are assigned zero costs.

We make sure that the port visit cost is incurred only once at each port even if a ship makes an over-night stay at the port. To account properly for the port visit cost in the time expanded network, we subtract the port visit cost for ship type $a$ at the port from the ship cost $c_e^{s,a}$ on the ground edge $e$. Thus if a ship makes an overnight stay at a port then though the port visit cost is counted twice via the node costs it is subtracted once via the ground edge cost. Idling of a ship at a port is penalized by imposing the overnight stay cost on the ship. Long stay of cargo at a port, which is different from its port of destination, is penalized through the holding cost at the port. In our model, we use days of the week as the level of time discretization, thus we assume that if ships on two different cycles meet at a port on the same day then transshipment can occur in both the directions (i.e. cargo can be transferred from both the ships on each other). By considering finer discretization of time, one can account for smaller time windows on which ships at a port meet for transshipments and to determine if transshipment is possible in only one direction (without incurring the holding cost).

Given that in the space-time network the level of discretization is in days, a commodity that becomes available at port $o$ on the $i$th day of the week is represented as a supply on vertex $v_{(o,i)}$ in the network. We assume that supply appears at vertex $o$ (for destination $d$) at the same day of week every week. We believe assuming that an average amount of demand arises at a port on a given day is reasonable in our context since we are considering a tactical model. Since the demand from week to week is taken to be the same for a carrier, the service characteristics may also remain the same every week. To this end, we assume that given any cycle, the weekly frequency on the cycle is maintained using ships from the same fleet. We characterize a cycle $C$ by the port rotation that it follows, the days of the week it visits each port in its

rotation, the type of ship that is used to service $C$ and the number of ships $L_C$ it takes to maintain a weekly frequency on $C$. Due to the indivisibility of ships, $L_C$ is integral by definition. A cycle is said to be *feasible* if it satisfies pre-specified rules in terms of the number of ships required to maintain a weekly frequency on the cycle and the number of ports visited. We denote the set of all feasible cycles for ship type $a$ by $\mathcal{C}^a$. Mathematically, for a cycle $C \in \mathcal{C}^{\dashv}$, $L_C = \left\lceil \frac{\sum_{e \in C} l_e^a}{7} \right\rceil$. Whenever necessary we represent a cycle $C$ by a sequence of vertices, for example a cycle from vertex $v_1$ to vertex $v_r$ via $v_2, \cdots, v_{r-1}$ is represented as $C = v_1 - v_2 \cdots v_r - v_1$. Cost of a cycle $C \in \mathcal{C}^a$ is denoted by $Cost_C$ and is calculated as $Cost_C = \sum_{v \in C} c_v^{s,a} + \sum_{e \in C} c_e^{s,a}$.

### 2.2.1 Mathematical Model

We now present a mixed integer programming formulation for the simultaneous ship scheduling and cargo routing problem. Our formulation has two sets of variables. First, for every feasible cycle $C$ we define a binary variable $x_C$. $x_C = 1$ if a weekly frequency is maintained on cycle $C$ and is 0 otherwise. The $x_C$ variables are taken to be binary rather than integer as the possibility of departing two ships of same type from a port following the same port calls on the same days of the week is highly unlikely.

Next, we define non-negative continuous variables representing the flow on edges. For each edge $e \in E_g \cup E_v$ and each triplet $(o, d, i) \in \Theta$ we define $f_e^{(o,d,i)}$ to denote the flow of commodity represented by $(o, d, i)$ on edge $e$. For a fictitious edge $e = (v_{(d,j)}, u_{(o,i)}) \in E_f$, a single flow variable, $f_e^{(o,d,i)}$, for commodity $(o, d, i)$, is defined since the flow of other commodities on this edge is not allowed. Note that, we let the flow variables to be continuous since adjusting for picking a fractional container does not influence the solution quality very much for the purposes of our tactical model.

Before presenting the model we summarize the following additional assumptions that we make. A1-A3 are for the clarity of exposition and in no way restrict the

usability of our model, however assumption A4 is more significant. (A1) Capacity on ground edges is assumed to be infinite i.e. we do not put any restriction on the amount of cargo that can be handled/stored at a port. (A2) We assume that all costs on cargo can be modelled via edge costs i.e. we set $c_v^c = 0 \; \forall v \in V$. (A3) We assume that all cargo is available in identical 1 TEU containers. Thus, $D^{(o,d,i)}$ represents the number of containers of a commodity required at port $d$ that become available at port $o$ on day $i$ of the week. (A4) Finally, we do not consider the costs involved with transferring cargo from one ship to another. As discussed before, cargo incurs holding costs whenever it stays at a port overnight. To account for transshipment costs correctly we need to distinguish between the capacity provided by different cycles on the same edge of the network. If the edges are duplicated for all the feasible cycles it will increase the size of the graph tremendously. Thus, it is hard to account correctly for transshipment costs if the network is not known (i.e. the cycles to be operated have not been selected). Since our aim is to consider the network design and cargo routing problems simultaneously and generate feasible cycles as a sub-problem, we ignore the transshipment costs for now. In Section 2.5.4 we present a computational study to discuss the effects of transshipment costs on cargo routing decisions once a set of cycles, to be operated by the given fleet, has been selected.

The simultaneous ship scheduling and cargo routing problem can be formulated as the following mixed integer program:

$(SSSCR)$ :

$$\max \sum_{(o,d,i)\in\Theta} \sum_{j=1}^{7} R^{(o,d,i)} f^{(o,d,i)}_{(v_{(d,j)},v_{(o,i)})} - \sum_{(o,d,i)\in\Theta} \sum_{e\in E} c_e^c f_e^{(o,d,i)} - \sum_{a\in\mathcal{A}} \sum_{C\in\mathcal{C}^a} Cost_C x_C \qquad (1)$$

such that

$$\sum_{e \in InEdges(v)} f_e^{(o,d,i)} - \sum_{e \in OutEdges(v)} f_e^{(o,d,i)} = 0 \quad \forall v \in V, \forall (o,d,i) \in \Theta \tag{2}$$

$$\sum_{(o,d,i) \in \Theta} f_e^{(o,d,i)} - \sum_{a \in \mathcal{A}} \sum_{\{C \in \mathcal{C}^a : e \in C\}} T^a x_C \leq 0 \quad \forall e \in E_v \tag{3}$$

$$\sum_{j=1}^{7} f^{(o,d,i)}_{(v_{(d,j)}, v_{(o,i)})} \leq D^{(o,d,i)} \quad \forall (o,d,i) \in \Theta \tag{4}$$

$$\sum_{C \in \mathcal{C}^a} L_C x_C \leq N^a \quad \forall a \in \mathcal{A} \tag{5}$$

$$x_C \in \{0,1\} \quad \forall C \in \mathcal{C}^a \, , \forall a \in \mathcal{A} \tag{6}$$

$$f_e^{(o,d,i)} \geq 0 \quad \forall e \in E, \forall (o,d,i) \in \Theta. \tag{7}$$

We now explain the above formulation. The objective function (1) maximizes the net profit by subtracting the sum of operating costs from the revenue generated. The first term in the objective function denotes the total revenue generated by transporting cargo between various origin and destination pairs. The second term captures the cost incurred by cargo during its routing from the origin port to the destination port. The third term denotes the total cost of operating ships on the selected cycles.

Constraint (2) is a flow balance constraint at every vertex of the space-time network. It ensures that the total flow into vertex $v$, of each commodity $(o,d,i) \in \Theta$, is equal to the total flow out of it for the same commodity. Constraints (3) and (4) are capacity constraints on the edges. Constraint (3) requires that the total flow on a voyage edge must be less than the sum of the capacities of ships servicing that edge. Constraint (4) models that the total flow of a given commodity from an origin port to a destination port must be less than the demand at the destination port. Note that we do not have a capacity constraint on ground edges because of assumption A1. Constraint (5) requires that for each fleet type, we do not use more ships than we have available. Note that if cycle $C \in \mathcal{C}^a$ is selected, i.e. $x_C = 1$, then it will utilize $L_C$ ships of type $a$ to maintain a weekly frequency. Finally, (6) denotes $x_C$ as binary variables and (7) denotes $f_e^{(o,d,i)}$ as non-negative continuous flow variables.

### 2.2.2 Hardness of the Problem

The decision version of the simultaneous ship scheduling and cargo routing problem is in NP, that is given a set of cycles to be operated and a flow of cargo on the edges it can be determined in polynomial time whether the total revenue generated is greater than a given constant K. We show the NP completeness of the problem by reducing a well known NP-complete problem, *0-1 Knapsack*, into a simultaneous ship scheduling and cargo routing problem.

The decision version of the 0-1 Knapsack problem is defined as the following: Given set $N = \{1, 2, \cdots n\}$, integers $K$, $c_i$ and $w_i$ for every $i \in N$ is there a subset $S$ of $N$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} c_i \geq K$?

**Theorem 1.** *The decision version of the simultaneous ship scheduling and cargo routing problem is NP-complete.*

*Proof.* Suppose there are $W$ identical ships with capacity $T$ TEUs each. Construct a sea cargo network as follows. For each $i \in \{1, 2 \cdots n\}$ construct two ports, a demand port($d_i$) with demand $c_i$TEUs and an origin port ($o_i$). Let a ship in the fleet take $\frac{w_i}{2}$ weeks to make a sailing from port $o_i$ to port $d_i$. Assume symmetric distances between ports and that the distances between $o_i$ and $d_j$, $\forall 1 \leq i \neq j \leq n$ is large. Thus $w_i$ ships are needed to maintain weekly frequencies on cycles $C_i = o_i - d_i - o_i$ for $1 \leq i \leq n$ and all other cycles are infeasible. Let $T = \max_{i \in N} c_i$ and let revenue generated by satisfying unit demand between any $o - d$ pair be 1. Assume there are no operating costs involved. Observe that :

- All feasible cycles are disjoint.
- A cycle $C_i$ needs $w_i$ ships to maintain weekly frequency and can generate $c_i$ units of revenue.

- If a set $S \subset N$ of cycles is chosen to be serviced then $\sum_{i \in S} w_i \leq W$ and a total of $\sum_{i \in S} c_i$ units of revenue is generated.

It follows easily now that a set of chosen cycles, $S$, will give a revenue of $K$ units or more if and only if the 0-1 Knapsack problem has a feasible solution. Thus the 0-1 Knapsack can be solved by solving a $SSSCR$ problem. □

## *2.3 Solution Methodology*

The linear program given by (1)-(7) contains a large number of variables even for moderate size problems. The large size of the model is a direct result of the exponential number of possible feasible cycles. Furthermore, each demand triplet adds a set of flow variables to the MIP model. An interesting observation however is that if we determine the set of cycles to be operated for each fleet type, i.e. given non-negative values $\overline{x_C}$ satisfying fleet availability constraints (5), model (1)-(7) reduces to the following *multicommodity flow problem* where each demand triplet is considered as a different commodity.

$$(MCF): \quad \max \sum_{(o,d,i)\in\Theta} \sum_{j=1}^{7} R^{(o,d,i)} f^{(o,d,i)}_{(v_{(d,j)},v_{(o,i)})} - \sum_{(o,d,i)\in\Theta} \sum_{e\in E} c^c_e f^{(o,d,i)}_e \tag{8}$$

such that

$$\sum_{e\in InEdges(v)} f^{(o,d,i)}_e - \sum_{e\in OutEdges(v)} f^{(o,d,i)}_e = 0 \quad \forall v \in V, \forall (o,d,i) \in \Theta \tag{9}$$

$$\sum_{(o,d,i)\in\Theta} f^{(o,d,i)}_e - \sum_{a\in\mathcal{A}} \sum_{\{C\in\mathcal{C}^a : e\in C\}} T^a \overline{x_C} \leq 0 \quad \forall e \in E_v \tag{10}$$

$$\sum_{j=1}^{7} f^{(o,d,i)}_{(v_{(d,j)},v_{(o,i)})} \leq D^{(o,d,i)} \quad \forall (o,d,i) \in \Theta \tag{11}$$

$$f^{(o,d,i)}_e \geq 0 \quad \forall e \in E, \forall (o,d,i) \in \Theta. \tag{12}$$

Note that (8)-(41) is a linear program with no integrality constraints as it only involves the flow variables $f^{(o,d,i)}_e$. Let $\pi = \{\pi^{(o,d,i)}_v : \pi^{(o,d,i)}_v \text{unrestricted}, \quad \forall v \in V, \ \forall (o,d,i) \in \Theta\}$ , $\lambda = \{\lambda_e : \lambda_e \geq 0 \quad \forall e \in E_v\}$ and $\omega = \{\omega^{(o,d,i)} : \omega^{(o,d,i)} \geq$

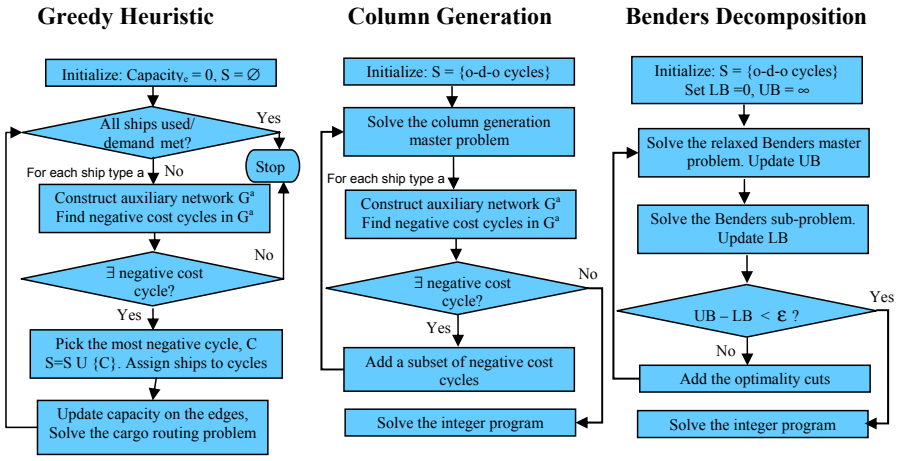


**Figure 5:** Outline of the three algorithms considered.

0 $\forall (o,d,i) \in \Theta\}$ be the set of dual variables associated with constraints (9), (10) and (11) respectively.

Next, we present three heuristic algorithms that exploit the above observation to solve the SSSCR problem. First, we provide a simple greedy heuristic that selects good cycles one by one and then assigns cargo to routes. Then we present a column generation based algorithm that generates a pool of good cycles and then selects the best cycles among these while also routing cargo in the network. Finally we present the details of a more involved Benders decomposition based algorithm. Figure 5 presents an outline of the three algorithms.

### 2.3.1 Greedy Algorithm

Let $S$ represent the set of cycles that are in operation i.e. ships have been assigned to maintain weekly frequencies on cycles in set $S$. The desirability or the value of cycle $C$ depends on the revenue generated by routing flow on ships employed in $C$, the number of ships required to maintain weekly frequency on $C$ and the various costs involved in operating the cycle $C$. The marginal value of cycle $C$ also depends on the cycles already present in set $S$. Thus, a greedy selection of cycles must take into account the set of existing operational cycles and demand triplets $(o,d,i) \in \Theta$.

Let $A^a$, $\forall a \in \mathcal{A}$, represent the number of ships of type $a$ that are currently available. The greedy algorithm starts with an empty set of selected cycles. To find profitable cycles an auxiliary network $G^a = (V^a, E^a)$ is created utilizing dual information from the solution of the $MCF$ problem to assign edge costs. $G^a$ is constructed for each ship type $a$ such that $V^a = V$ and $E^a = E^a_v \cup E^a_g$. Each edge $e \in E^a$ is assigned a cost $c_e = c^{s,a}_e + \lambda_e$ and each vertex $v \in V^a$ is assigned a cost $c_v = c^{s,a}_v$. For every ship type, the algorithm then finds a minimum cost cycle in the auxiliary network by using a procedure $FindCycle(G^a)$. Details of this procedure will be provided in Section 2.4. Finally, if feasible the cycle with minimum cost is selected and a suitable number of ships, to maintain weekly frequency, is assigned to it. The process is repeated while there are ships to be assigned.

### 2.3.2 Column Generation Based Algorithm

Though the greedy algorithm is simple and provides a feasible solution quickly it is not very effective. It works with a very small set of feasible cycles and once a feasible cycle is generated it is picked in the final solution without any further considerations. Next we propose a column generation based algorithm that iteratively generates a good pool of profitable cycles for solving the linear programming (LP) relaxation of (1)-(7).

Column generation is an effective way of solving linear programs with a large number of columns (see [11] for an introduction). Rather than enumerating all the columns explicitly, it begins by solving a restricted problem (called the master problem) with a select set of columns. A subproblem is solved to generate "attractive" columns and they are subsequently added to the master problem. The process is repeated until no further profitable columns can be generated. The column generation technique has been successfully used to solve many large scale optimization problems, please see ([7]) for an example in solving airline crew assignment problems.

To solve the LP relaxation of $SSSCR$, the master problem in the column generation is initialized by restricting the set of cycle selection variables, to one simple cycle for every demand triplet. At every step of the column generation process, the master problem is solved to find the best value for all the decision variables. The pricing subproblem for the column generation is equivalent to identifying negative cost cycles in an auxiliary network, for every ship type. The auxiliary network, $G^a = (V^a, E^a)$ is constructed for each ship type $a$ such that $V^a = V$ and $E^a = E^a_v \cup E^a_g$. Dual variable values from the master problem are used to assign costs to the edges and the vertices of the auxiliary network. Each edge $e \in E^a$ is assigned a cost $c_e = c^{s,a}_e + T^a\lambda_e + \frac{l_e}{7}\sigma^a$ and each vertex $v \in V^a$ is assigned a cost $c_v = c^{s,a}_v$ so that negative cost cycles in $G^a$ correspond to columns with positive reduced costs in the master problem. Note that since $SSSCR$ is a maximization problem, profitable columns are the ones with positive reduced cost. Procedure $FindCycle(G^a)$ (described in Section 2.4) is used to identify negative cost cycles in the auxiliary network and corresponding columns are added to the master problem. The process is continued until no new cycles can be found. Finally, integrality constraints are imposed on the cycle selection variables and a branch-and-bound framework is used to obtain an integer solution for the $SSSCR$ problem. No new columns are generated during the branch-and-bound phase. Different branching rules, with different advantages, can be devised to obtain the integer solution. For example branching on the largest feasible cycle forces many other binary variables also to satisfy the integrality constraints. However, we use the variable that affects the solution quality the most as the variable to branch on, giving preference to the up ($x_C = 1$) branch, because this strategy performed the best in our computational experiments.

### 2.3.3 Benders Decomposition Based Algorithm

As the number of ports, ships and demand triplets increase, solving model (1)-(7) by column generation becomes increasingly difficult. The number of constraints as well as the number of variables increase with the increase in the number of ports, ships and demand triplets, in the column generation master problem. We next decompose the LP relaxation of the model (1)-(7) using Benders decomposition to obtain a pair of problems that utilize the separability of SSSCR. The decomposition results in a master problem, where the number of variables increases as the number of cycles increase, and a subproblem, where the number of constraints increases as the number of demand triplets increase. Thus the effect of the increase in problem size is divided between a master problem and a subproblem. Further, this decomposition is used to effectively solve the LP relaxation and the solution is embedded in a branch-and-bound approach to obtain an integer solution.

Benders decomposition ([10]) is a popular technique to solve mixed integer linear programming problems with linking constraints. This approach is useful when master problem has all the integer variables and it is difficult to treat them in sub-problems. The solution process iterates between an integer master problem, which passes on the value of integer variables to subproblem(s), and subproblems generate cuts (feasibility and optimality) which are passed back to the master problem. Though this approach has proved to be suitable for many problems it has the drawback that an integer master problem has to be solved at each iteration. [43] proposed a modification to this approach in which the solution of a sequence of integer programs is replaced by the solution of a sequence of linear programs and a few integer programs.

The basic techniques of [43] and its modifications have been used successfully to solve many hard problems. [28] used it for solving an engine scheduling problem and [66] used it for solving a time-dependent travelling salesman problem. More

recently, these techniques have been used successfully to solve locomotive car assignment ([17], [18]) and aircraft routing and crew scheduling ([19]) problems. For these problems enormous time reductions and significant improvements in solution quality were achieved by first relaxing the integrality constraints in the master problem. After the relaxation is solved, to acceptable time or optimality criteria, the integrality constraints are introduced back in the master problem. We now present the use of Benders decomposition method to solve the $SSSCR$ problem.

#### *2.3.3.1 Benders reformulation*

As noted earlier for given non-negative values $\overline{x_C}$ satisfying fleet constraints (5), the LP relaxation of model (1)-(7) reduces to the $MCF$. Since $MCF$ is a multicommodity flow problem with no integrality constraints, the optimal value of $MCF$ problem is equal to the optimal value of its dual. The dual problem ($DP$) of the $MCF$ problem can be written as

$$(DP): \quad \min \sum_{e \in E_v} \sum_{a \in \mathcal{A}} \sum_{\{C \in \mathcal{C}^a : e \in C\}} T^a \overline{x_C} \lambda_e + \sum_{(o,d,i) \in \Theta} D^{(o,d,i)} \omega^{(o,d,i)} \tag{13}$$

such that

$$\pi^{(o,d,i)}_{head(e)} - \pi^{(o,d,i)}_{tail(e)} + \lambda_e \geq -c^c_e \quad \forall e \in E - E_f \ , \ \forall (o,d,i) \in \Theta \tag{14}$$

$$\pi^{(o,d,i)}_{head(e)} - \pi^{(o,d,i)}_{tail(e)} + \omega^{(o,d,i)} \geq R^{(o,d,i)} - c^c_e \quad \forall e \in E_f \ , \ \forall (o,d,i) \in \Theta \tag{15}$$

$$\pi^{(o,d,i)}_v \quad \text{unrestricted}, \quad \forall v \in V, \ \forall \ (o,d,i) \in \Theta \tag{16}$$

$$\lambda_e \geq 0 \quad \forall e \in E_v \tag{17}$$

$$\omega^{(o,d,i)} \geq 0 \quad \forall \ (o,d,i) \in \Theta. \tag{18}$$

Let $D$ be the feasible region of the dual problem and $P_D$ and $Q_D$ be the set of extreme points and extreme rays of $D$, respectively. Note that $D$ does not depend on $\overline{x_C}$. Also, since $R^{(o,d,i)} \geq 0 \ \ \forall (o,d,i) \in \Theta$ and $c^c_e = 0 \ \forall e \in E_f$, a feasible solution for the dual subproblem is $\pi^{(o,d,i)}_v = 0 \ \forall \ v \in V, \ \forall (o,d,i) \in \Theta, \ \lambda_e = 0 \ \forall \ e \in E_v$

and $\omega^{(o,d,i)} = R^{(o,d,i)}\ \forall\ (o,d,i) \in \Theta$, and thus $D \neq \emptyset$. Now, by strong duality, either the $MCF$ problem is infeasible or it is feasible and bounded. Clearly the null vector $0$ is a feasible solution for $MCF$. This means that the primal-dual pair of $MCF$ and $DP$ is feasible and bounded. Thus the optimal value of $MCF$ and $DP$ can be characterized in terms of only the extreme points of $DP$, i.e. the set $P_D$, and can be written as:

$$\min_{(\pi,\lambda,\omega)\in P_D} \sum_{e\in E_v} \{\sum_{a\in\mathcal{A}} \sum_{\{C\in\mathcal{C}^a:e\in C\}} T^a\overline{x_C}\}\lambda_e + \sum_{(o,d,i)\in\Theta} D^{(o,d,i)}\omega^{(o,d,i)}.$$

Introducing an additional free variable $z$, model (1)-(7) can be reformulated as the following *Benders master problem* (BMP). This problem has integer variables $x_C$ and one free continuous variable $z$.

$$(BMP): \quad \max z \tag{19}$$

such that

$$z \leq \sum_{e\in E_v} \{\sum_{a\in\mathcal{A}} \sum_{\{C\in\mathcal{C}^a:e\in C\}} T^a x_C\}\lambda_e + \sum_{(o,d,i)\in\Theta} D^{(o,d,i)}\omega^{(o,d,i)} - \sum_{a\in\mathcal{A}} \sum_{C\in\mathcal{C}^a} Cost_C x_C \ \ \forall(\lambda,\omega)\in P_D \tag{20}$$

$$\sum_{C\in\mathcal{C}^a} L_C x_C \leq N^a \quad \forall\ a\in\mathcal{A} \tag{21}$$

$$x_C \in \{0,1\} \tag{22}$$

$$z \quad free. \tag{23}$$

Note that we do not have any feasibility constraints in the Benders master problem because (DP) is bounded. The optimality constraints (20) ensure that $z$ is restricted to be smaller than or equal to the value of the right hand side of constraint (20) at various extreme points of $DP$. In general, the above model contains many more constraints than the LP relaxation of model (1)-(7) but most of them are inactive at optimality. Thus a natural approach to solve (19)-(23) is by dropping constraints (20) and generating them as needed. We now present the basic Benders algorithm

to solve the linear relaxation of $SSSCR$ problem to optimality. Later, integrality constraints are introduced to solve the original $SSSCR$ problem. We denote the linear relaxation of $BMP$ as $LPBMP$ and the relaxation of $LPBMP$ obtained by dropping constraints (20) as the $RLPBMP$.

#### *2.3.3.2 Overview of the algorithm*

The basic Benders decomposition based algorithm for solving the LP relaxation of $SSSCR$ iteratively selects good cycles, by solving the $RLPBMP$, for the ship scheduling problem and then efficiently solves the cargo routing problem, by solving the $MCF$ problem. The $MCF$ problem utilizes the $RLPBMP$ solution to assign capacity to voyage edges before solving the flow problem. In return, at each iteration, the dual solution of the $MCF$ problem provides an optimality cut to the $RLPBMP$.

Let $t$ be the iteration number and $P_D^t$ be the restricted set of extreme points of $D$ available at iteration $t$, i.e. the $RLPBMP$ at iteration $t$ is obtained from $LPBMP$ by replacing $P_D$ by $P_D^t$ in (20). Note that the solution of the $RLPBMP$ at each iteration $t$, denoted by $z^t$, provides an upper bound for the original $LPBMP$ (since the $RLPBMP$ has fewer constraints than the $LPBMP$).

**Algorithm 1** The basic Benders decomposition based algorithm

Procedure Basic Benders()

Set $t = 1$, $P_D^t = \emptyset$, $lower\_bound = 0$, $upper\_bound = \infty$.

**while** ($upper\_bound > lower\_bound + \epsilon$ ) **do**

STEP 1. $SOLVE$ the $RLPBMP$ to obtain solution $z^t$ and $\{\overline{x_C}\}^t$.

Set $upper\_bound = z^t$.

STEP 2. Solve the $MCF$ problem taking $\{\overline{x_C}\}^t$ as input to obtain $v(\{\overline{x_C}\}^t)$ and optimal dual solution $(\overline{\pi}, \overline{\lambda}, \overline{\omega})$.

Set $lower\_bound = \max\{lower\_bound, v(\{\overline{x_C}\}^t) - \sum_{a\in\mathcal{A}} \sum_{C\in\mathcal{C}^a} Cost_C \overline{x_C}\}$.

Set $P_D^{t+1} = \{P_D^t \cap \{(x,z) : z \leq \sum_{e\in E_v} \{\sum_{a\in\mathcal{A}} \sum_{\{C\in\mathcal{C}^a : e\in C\}} T^a x_C\}\lambda_e + \sum_{(o,d,i)\in\Theta} D^{(o,d,i)}\omega^{(o,d,i)} - \sum_{a\in\mathcal{A}} \sum_{C\in\mathcal{C}^a} Cost_C x_C\}\}$.

$t = t + 1$.

**end while**

The original $BMP$ (or $SSSCR$) problem with integrality constraints is solved heuristically in two phases. In Phase I, all integrality constraints are relaxed and the $LPBMP$ is solved to optimality by using basic Benders decomposition algorithm (Algorithm 1). Since the set of feasible cycles can be exponential, the $RLPBMP$, in this phase, is solved in a column generation setting. For phase I, $SOLVE$ in Algorithm 1 refers to this column generation and its details will be provided next.

Retaining all optimality cuts and cycles generated in the first phase, Phase II puts the integrality constraints back on the master problem. Algorithm 1 is started once more, however in this phase the $RLPBMP$ in Step 1 is replaced with the mixed integer program $BMP$, together with the cuts and cycles generated in the first phase. Since the $DP$ polytope is not affected by the integrality constraints, all optimality cuts generated in Phase I can be used to generate corresponding cuts for the mixed integer program in Phase II. Additional optimality cuts are generated at each iteration. Note however that in phase II Algorithm 1 in Step 1 solves an integer problem at every iteration. Thus in Phase II no new cycles are generated and $SOLVE$ simply refers to a branch-and-bound solution of the relaxed $BMP$. This two phase approach for solving integer programs using Benders decomposition was originally proposed by [43] and the intuition behind it is the hope that many of the necessary constraints for the master problem may be generated by solving a linear program in place of the more computationally expensive integer program.

The branch-and-bound tree in Phase II is searched by a depth first search, giving preference to the up ($x_C = 1$) branch. As in Section 2.3.2, the variable that affects the solution quality the most is chosen as the branching variable. Note that solving the mixed integer program in $BMP$ is a computationally expensive step. Since any feasible integer solution can be used to generate an optimality cut, the mixed integer program $BMP$ does not need to be solved to optimality at every iteration. However, if the $BMP$ is solved heuristically the upper bound it provides during the Benders

iterations could be much smaller than the true upper bound which might lead to a premature termination of the algorithm. In the worst case, the upper bound could become smaller than the lower bound. To avoid such a premature termination of the algorithm, the branch-and-bound search is terminated only when the solution quality obtained reaches an acceptable optimality gap (the gap between the best integer objective and the objective of the best node remaining). Searching the branch-and-bound tree for a solution with small optimality gap is likely to take large computation time but it is also likely to provide better solution quality by providing better bounds for the Benders iterations. Thus, a suitable optimality gap must be chosen to avoid the premature termination of the algorithm and to keep it computationally efficient.

#### *2.3.3.3 Column generation for solving the RLPBMP*

The master problem in the Benders decomposition (the $RLPBMP$ in Algorithm 1) is solved in a column generation setting. The pricing subproblem in the column generation reduces to identifying negative cost cycles in an auxiliary network, $G^a = (V^a, E^a)$, for every ship type $a$. As before, $G^a$ is constructed such that $V^a = V$ and $E^a = E^a_v \cup E^a_g$. We next present how we compute the costs on the vertices and the edges of network $G^a$.

Let $\Pi_{(\lambda,\omega)}$ and $\sigma^a$ denote the dual variables corresponding to constraint (20) and (21), respectively. The reduced cost $\overline{c_C}$ of a cycle $C \in \mathcal{C}^a$ can now be written as:

$$\overline{c_C} = 0 - \left( \sum_{(\lambda,\omega)} \left( Cost_C - \sum_{\{e:e\in C\cap E_v\}} T^a \lambda_e \right) \Pi_{(\lambda,\omega)} \quad + \quad L_C \sigma^a \right). \tag{24}$$

Note that in (24) the second summation is only over the voyage edges of cycle $C$. Since the ground edges have infinite capacity, at optimality, by complementary slackness conditions $\lambda_e = 0 \; \forall e \in E_g$. Thus, ground edges can also be included in the summation in (24). From LP theory, we know that if the reduced cost $\overline{c_C} \leq 0$ for each cycle $C \in \mathcal{C}^a$ and every fleet type $a \in \mathcal{A}$ then we have the optimal solution to

our problem. That is the column generation iterates as long as there exists a cycle $C \in \mathcal{C}^a$ for some $a \in \mathcal{A}$ such that

$$\sum_{(\lambda,\omega)} \left( \sum_{v \in C} c_v^{s,a} + \sum_{e \in C} c_e^{s,a} - \sum_{e \in C} T^a \lambda_e \right) \Pi_{(\lambda,\omega)} \quad + \quad \left\lceil \sum_{e \in C} \frac{l_e^a}{7} \right\rceil \sigma^a < 0$$

$$\Rightarrow \sum_{v \in C} \left( \sum_{(\lambda,\omega)} \Pi_{(\lambda,\omega)} \right) c_v^{s,a} - \sum_{e \in C} \left( \sum_{(\lambda,\omega)} \Pi_{(\lambda,\omega)} * \lambda_e \right) T^a$$

$$+ \sum_{e \in C} \left( \sum_{(\lambda,\omega)} \Pi_{(\lambda,\omega)} \right) c_e^{s,a} \quad + \quad \left\lceil \sum_{e \in C} \frac{l_e^a}{7} \right\rceil \sigma^a < 0. \tag{25}$$

For the network $G^a$, we assign cost $c_v = \left( \sum_{(\lambda,\omega)} \Pi_{(\lambda,\omega)} \right) c_v^{s,a}$ to every $v \in V^a$ and cost $c_e = \frac{l_e}{7}\sigma^a \; - \; \left( \sum_{(\lambda,\omega)} \Pi_{(\lambda,\omega)} * \lambda_e \right) T^a + \left( \sum_{(\lambda,\omega)} \Pi_{(\lambda,\omega)} \right) c_e^{s,a}$ to every edge $e \in E^a$. Let, $Cost_C$ represents the cost of cycle $C$ with the above cost structure. Let $\lceil Cost_C \rceil$ be the cost when we replace the $\sum_{e \in C} \frac{l_e^a}{7}$ term in $Cost_C$ by $\left\lceil \sum_{e \in C} \frac{l_e^a}{7} \right\rceil$. Note that, since $\lceil x \rceil \geq x$, if the optimal value of the pricing subproblem $\forall \; a \in \mathcal{A}$ is greater than zero then there are no more profitable cycles because $Cost_C \geq 0 \Rightarrow \lceil Cost_C \rceil \geq 0$. If however, the optimal value of the pricing subproblem for some $a \in \mathcal{A}$ is less than zero than we need to check if $\lceil Cost_C \rceil < 0$. If it is, then we have found a profitable cycle, otherwise either there are no more profitable cycles to be added or profitable cycles have very low negative cost ($> -1$) and are therefore ignored. We use Procedure $FindCycle(G^a)$ (as will be described in Section 2.4) to identify negative cost cycles in $G^a$.

## *2.4 Algorithmic Issues*

In this section we discuss several algorithmic ideas we utilize to make our algorithms more effective, efficient, and stable.

### 2.4.1 Solving the Pricing Subproblem

The pricing subproblems for both the column generation and Benders decomposition based algorithms reduce to finding profitable cycles in the auxiliary network $G^a$. Similarly, the Greedy algorithm needs to find profitable cycles in the auxiliary network. It is tempting to solve directly a minimum cost circulation problem in the network $G^a$, to identify negative cost cycles. However, the cycles obtained by decomposing the solution of the circulation problem into simple cycles are not guaranteed to be practical. For example, our initial computational experiments with the circulation problem suggest that most of the cycles thus generated are too long and require a large number of ships to maintain weekly frequency. Hence, we first discuss rules based on the real world practice of liner shipping companies for defining feasible cycles. Next, a recursive algorithm, $FindCycle(G)$, is presented to efficiently find negative cost cycles, satisfying pre-defined feasibility conditions, in a given network $G$.

#### *2.4.1.1 Defining feasible cycles*

To solve the pricing subproblem to optimality one must consider all sequences of ports as candidates for possible profitable cycles. However, searching for negative cost cycles in such an unconstrained manner not only makes our algorithms inefficient by generating cycles that create undesirable effects such as large integrality gaps but also it generates cycles that would never be operated in practice. Hence we impose a set of constraints that a cycle must satisfy to qualify as a feasible cycle.

Global carriers operate in different regions, for example OOCL operates mainly in North America, Europe and Asia, and cater to the demand of various markets, such as trans-Atlantic, trans-Pacific, intra-Asia, and Asia-Europe trade routes. Figure 6 represents an Asia-Europe cycle for OOCL. For a carrier it is important to tap the benefits of both inter-region and intra-region markets. Whereas, some of the inter-region markets, for example the trans-Pacific, are the most profitable ones, some of


**Figure 6:** An Asia-Europe cycle for OOCL. Source: OOCL

the intra-region markets, for example the intra-Asia market, form the backbone of international shipping.

We considered the published cycles by [47] and [6] in trans-Pacific and intra-Asia trade routes to come up with the following guidelines for defining the set of feasible cycles distributed in two regions, $r_i$ and $r_j$.

1. The number of ports visited by a cycle must not be too high. Most of the current trans-Pacific cycles visit up to 10-15 ports and intra-Asia cycles visit up to 7-10 ports. Let $R_{(r_i,r_j)}$ denote the maximum number of ports that a cycle visiting region $r_i$ and $r_j$ is allowed to visit.

2. The length (in weeks) of a cycle must be bounded by a suitable number, i.e. the number of ships that can be committed to a particular cycle are limited. Most of the trans-Pacific cycles are up to 15 weeks long and most of the intra-Asia cycles are up to 6 weeks long. Let $L_{(r_i,r_j)}$ be the maximum allowed length in weeks for a cycle visiting region $r_i$ and $r_j$.

3. Cycles that operate in multiple regions must enter and leave a region only once, i.e. no inter-region loops are allowed. However, 1-2 intra-region loops are allowed.

4. Each cycle must directly (without using capacity on other cycles) serve the origin and destination ports of at least one demand triplet. It is highly unlikely

for a carrier to introduce a cycle that does not satisfy any demand directly.

#### *2.4.1.2 Finding negative cost feasible cycles*

Incorporating any of the rules that guide the feasibility of a cycle into the circulation problem yields a NP hard problem. (This is easily seen from the fact that shortest weight-constrained path problem is NP complete ([29]).) Furthermore, an exhaustive enumeration of cycles following the above rules still yields a large number of cycles. For ports distributed in two regions, up to 10,000 cycles for a 10 port, 30 demand triplets problem, over a million cycles for a 15 port, 50 demand triplets problem and more than 10 million cycles for a 20 port, 80 demand triplets problem exist.

We now describe an iterative search algorithm for constrained negative cycle detection which yields good computational results. In essence, the algorithm utilizes Lemma (1) due to [40] to prune the search tree by ignoring paths with non-negative costs. This pruning helps the algorithm to maintain time- and space-efficiency. [4] have used lemma (1) to develop a similar algorithm for detecting subset disjoint negative cycles.

**Lemma 1.** *For a negative cost (directed) cycle* $C = v_1 - v_2 - .... - v_r - v_1$ *there exists a node* $v_h$ *in* $C$ *such that each partial (directed) path* $v_h - v_{h+1}$, $v_h - v_{h+1} - v_{h+2}$, $v_h - v_{h+1} - v_{h+2} - ...$ *(where indices are modulo r) is a negative cost (directed) path.*

We now present a cycle generation algorithm for ports distributed in two regions: $r_1$ and $r_2$. Note that these ideas can easily be carried over to ports distributed in more than two regions. Before presenting the algorithm we define some notation. With each directed path $p$, we associate the following information: $head(p)$ and $tail(p)$ denote the last node and first node on $p$. $Cost(p)$ denotes the cost of path $p$. $NR_{r_1}(p)$ and $NR_{r_2}(p)$ denote the number of ports from region $r_1$ and $r_2$, $ER(p)$ denotes the number of inter-region edges and finally $l(p)$ denotes the length of path $p$. Note that each edge in the network is either between two nodes of the same region (intra-region

edge) or between two nodes of different regions (inter-region edge) and that the set $\{NR_{r_1}(p), NR_{r_2}(p), ER(p)\}$ completely describes the region(s) visited by a path $p$. For a path $p$ we denote the set $\{head(p), tail(p), ER(p)\}$ as $DSet(p)$. We say that a path $p$ *dominates* another path $q$ if $DSet(p) = DSet(q)$ and $Cost(p) < Cost(q)$.

Note that a cycle can be obtained by connecting the endpoints of a path. Lemma (1) suggests that to find negative cost cycles it is enough to consider paths with negative cost. Further, the above definition of dominance suggests that among the paths with the same $DSet()$ only the path with the least cost needs to be explored further. A path $p$ is said to be feasible if it has negative cost and if it can be extended to form a feasible cycle. Let $P_k$ denote the set of all non-dominated, feasible paths with k nodes.

For each ship type $a$, Algorithm (2) detects negative cost cycles in the auxiliary network $G^a$, described earlier with various cost structures. It works inductively by constructing set $P_{k+1}$ from the set $P_k$. For each path $p \in P_k$ it examines if the path can be extended by adding a single edge to form path $p'$, that is if path $p'$ is feasible. Procedure $if\_feasible\_path(p)$ checks for the feasibility of path $p$ depending on the region(s) visited by $p$, by ensuring that the guidelines set in Section 2.4.1.1 are met, and accounts for the fact that for a ship type $a$ no cycle can be longer than $N^a$, number of available ships for ship type $a$, weeks. The path is then checked for dominance in $P_{k+1}$, using procedure $if\_dominated(p', P_{k+1})$ and non-dominated paths are added to $P_{k+1}$. For a cycle $C$, procedure $if\_feasible\_cycle(C)$ checks if the cycle $C$ is feasible.

For a path $p$ all information can be maintained in $O(1)$ time. For example, to maintain $ER(p)$, we assign a value 0 to all intra-region edges and value 1 to inter-region edges. Thus whenever an edge is appended to a path $p \in P_k$ to obtain path $p' \in P_{k+1}$, $ER(p')$ can be obtained by adding the value of the appended edge to $ER(p)$. Since we maintain all information regarding the region(s) visited by a path, feasibility check for a path and a cycle can be done in constant time. To check the

dominance of a path $p \in P_k$, we first need to check if there exists a path $q \in P_k$ such that $DSet(p) = DSet(q)$. This is a computationally expensive step. We use standard hashing techniques to efficiently detect paths with the same $DSet$. Once such a path is found, dominance can be checked in $O(1)$ time. Note that at any given time, set $P_k$ will contain only one path with a particular $DSet$, i.e. the non-dominated path. Rather than storing only the most negative cycle $C^*$ Algorithm (2) can easily be modified to maintain a pre-defined number of best cycles.

**Algorithm 2** An iterative constrained negative cycle detection algorithm

Procedure FindCycles($G^a$)
  **for all** $e \in E^a$ **do**
    $p = \{e\}$
    **if** $if\_feasible\_path(p)$ **then** $P_1 = P_1 \cup \{p\}$
  **end for**
  $k = 1$, $C^* = \emptyset$, $Cost_{C^*} = 0$
  **while** $k < R$ **do**
    **while** $P_k \neq \emptyset$ **do**
      Remove a path $p$ from $P_k$
      Connect the ends of path $p$ to form cycle $C$.
      **if** $if\_feasible\_cycle(C)$ and $Cost_C < Cost_{C^*}$ **then** $C^* = C$
      **for all** $\{(head(p), j) \in OutEdges(head(p))\}$ **do**
        $p' = p \cup \{(head(p), j)\}$
        **if** $\{if\_feasible\_path(p')\}$ **then**
          $P_{k+1} = P_{k+1} \cup \{p'\}$
          **if** $if\_dominated(p', P_{k+1})$ **then** Remove the dominated path
        **end if**
      **end for**
    **end while**
    k+1
  **end while**

### 2.4.2 Choosing an Initial Set of Cuts

Even though the Benders decomposition based Algorithm (1) may be initialized with an empty set of extreme points, the choice of an initial set may affect its convergence. In our experiments, the addition of several cuts helped us improve the performance of Algorithm (1).

Note that, $\pi_v^{(o,d,i)} = 0\ \forall v \in V,\ \forall (o,d,i) \in \Theta$, $\lambda_e = 0\ \forall e \in E$ and $\omega^{(o,d,i)} = R^{(o,d,i)}$ $\forall (o,d,i) \in \Theta$ is a feasible but not necessarily an extreme point solution of the $DP$ polytope. It can thus be used to obtain the valid cut

$$z \leq \sum_{(o,d,i)} R^{(o,d,i)} D^{(o,d,i)} - \sum_{a \in \mathcal{A}} \sum_{C \in \mathcal{C}^a} Cost_C x_C. \tag{26}$$

The above cut is equivalent to adding the constraint that the value of the optimal solution must be less than or equal to the revenue that can be generated by satisfying all the available demand minus the cost of operating the picked cycles.

Similarly, $\lambda_e = \max_{(o,d,i) \in \Theta} R^{(o,d,i)}\ \ \forall e \in E$ and $\omega_{(o,d,i)} = 0 \forall (o,d,i) \in \Theta$ is a feasible solution for the $DP$ polytope and provides the valid cut

$$z \leq \max_{(o,d,i) \in \Theta} R^{(o,d,i)} \sum_{e \in E_v} \left\{ \sum_{a \in \mathcal{A}} \sum_{C \in \mathcal{C}^a : e \in C} T^a x_C \right\} - \sum_{a \in \mathcal{A}} \sum_{C \in \mathcal{C}^a} Cost_C x_C. \tag{27}$$

### 2.4.3 Making Column Generation Effective

While performing column generation, both in the pure column generation based algorithm for $SSSCR$ and for solving the $RLPBMP$ in Benders decomposition based algorithm, we identify and add more than one profitable column per iteration. During the iterative cycle generation instead of maintaining just the most negative cycle we maintain a set of $5 - 10$ most profitable cycles at almost no extra cost. This helps to significantly reduce the numbers of iterations during column generation without substantially increasing the time taken per iteration.

During a typical column generation, the problem keeps growing as the column generation process keeps adding columns to the master problem. To keep the list of columns manageable, we frequently delete nonbasic columns with high negative reduced cost from the master problem. This reduces the time per iteration significantly, though it increases the number of iterations slightly in many cases. As another speed up for the column generation process, if no new cycle is detected for a ship type in an iteration then cycle generation for that ship type is suspended for 2-5 iterations.

## *2.5 Computational Experiments*

In this section, we present the results of our computational study after describing the schema employed for generating test cases. We first establish the dominance of the Benders decomposition based algorithm over the greedy heuristic and the column generation based algorithm. Next, we present a deeper analysis of the Benders decomposition based algorithm. Finally, we discuss some of the interesting characteristics of the solutions obtained by our algorithm and show that it supports the recent trends observed in the sea-cargo industry. All of our algorithms were implemented in C++ in an Unix environment and we made extensive use of the callable libraries in CPLEX 9.0. All computational experiments were performed on a Sun280R workstation with UltraSparc-III processor. All times are reported in minutes.

### 2.5.1 Data Generation

We performed our computational experiments on networks with ports distributed in two regions. Each generated port is randomly assigned to one of the two regions, with equal probability, and the sailing distance between ports are chosen to represent the sailing distance between ports distributed in the Asia and the North-America regions. Typically, as observed from [47] and [6] service networks, intra-region sailing times for ports in Asia and North-America are 2-30 days whereas the inter-region sailing times are 14-42 days.

Origin-destination pairs are chosen randomly from the pairs of ports. Day of the week on which supply arises at the origin port is assumed to be the same every week and is chosen uniformly at random from the seven days of a week. The demand sizes are randomly generated from the interval 0.1 to 1.0 times the capacity of the largest ship available. Similar proportions are used in [27] and it is suggested that this represents the demand sizes observed by a liner shipping company. Revenue generated by satisfying demand for a given demand triplet $(o, d, i)$ is chosen to be in direct

proportion to the distance between port $o$ and port $d$, i.e. more revenue is generated by satisfying a demand at a port in North-America from a port in Asia as compared to satisfying a demand between two ports in North-America. The proportionality constant is chosen randomly from $[100, 200]$.

Since the fleet of a carrier usually consists of ships of different types we considered three different ship types in our fleet. The three ship types have capacity 2000 TEU, 4000 TEU and 8000 TEU. [9] and [35] suggest that ships with 2000 TEU and 4000 TEU capacity are currently in use. According to [47], OOCL has ships of different types with capacity varying from 2,500TEU to 8,063TEU. A recent increase in literature regarding the viability of larger ships, [35], points towards the increasing use of big ships and [9] suggests that ships of up to 8000+ TEUs are in design.

There are various fixed and variable costs involved in shipping a cargo. As in [16] we do not consider the daily running costs including cost of capital, personnel, insurance etc. since they are fixed during the planning period. However, we consider various operational costs that effect a carrier's decision regarding which ports to visit and which cycles to operate on. For every ship type, $a \in \mathcal{A}$, and for all the ports, $v \in \mathcal{P}$, we consider a port visit cost incurred by a ship of type $a$ if it visits port $v$. At a port $p$, port visit cost for a ship is proportional to the capacity of the ship i.e. a ship with 8000 TEU capacity incurs a higher port visit cost as compared to a 2000 TEU capacity ship. At every port, $v \in \mathcal{P}$, we consider a per unit cargo per night holding cost. This cost is incurred by a unit of cargo if it is held at a port for one night and is assumed to be the same for all cargo types. At a port, holding cost per unit cargo is chosen to be considerably smaller than the port visit cost for a ship. For every ship type, $a \in \mathcal{A}$, and for every pair of ports, $\{u, v\}$, we consider the operation cost for sailing a ship from port $u$ to $v$. The operation cost depends on the type of ship that is used for the sailing and is proportional to the distance between the ports.

We generate various classes of random instances, utilizing the above schema, to

test the robustness of our algorithm. Classes are characterized by specifying the number of ports ($P$), the number of ships ($S$) and the number of demand triplets ($D$). For example, an instance with 6 ports, 30 ships and 18 demand triplets is represented as $P6S30D18$. We tested our algorithm on networks with 6, 10, 15 and 20 ports to be serviced. In each of the test classes 20-30 % of all pairs of ports are considered to be origin destination pairs. A fleet size of up to 100 ships is scheduled. Grand Alliance which is one of world's largest alliances has a fleet of 100 ships and [6] has a fleet of more than 80 container ships. For each test class, results reported in this section were obtained by generating 5 random instances and then taking an average over them.

To report the results of our computational study in tabular form we use the following abbreviations:

- $G$: The greedy algorithm.
- $C$: The pure column generation based algorithm.
- $B$: The two phase Benders decomposition based algorithm where column generation is used for solving the master problem in Phase I.
- $F$: The cycle generation algorithm based on the flow decomposition of the circulation problem.
- $I$: The cycle generation algorithm based on the iterative search algorithm.

Combination of these are used to represent the overall algorithm tested. For example, the two phase Benders decomposition based algorithm with the iterative search algorithm for cycle generation is represented by $BI$.

### 2.5.2 Effectiveness of the Algorithms

We now compare the Benders decomposition based algorithm with the other proposed algorithms. While solving the problem with the pure column generation based

algorithm the LP relaxation is first solved to optimality and then the integer solution is obtained using branch-and-bound. However, while solving the problem with the two phase Benders decomposition based algorithm, in Phase I the cuts are generated until the relative difference between the upper bound provided by the Benders relaxed master problem and the lower bound provided by the subproblem is less than 1% or the number of iterations in the first phase of Benders are less than 200. Phase I terminates when one of these criteria is met. The LP solution obtained by the pure column generation based algorithm is used as an upper bound to estimate the quality of the final integer solution.

Table 6 presents a comparison between the greedy algorithm, the pure column generation based algorithm and the Benders decomposition based algorithm. It also compares the flow decomposition based cycle generation algorithm with the iterative search algorithm for cycle generation. The second and third column of Table 6 report the number of cycles generated and the CPU time taken to solve the problem using greedy algorithm with iterative cycle generation. The fourth and fifth column report these statistics for the pure column generation based algorithm with iterative cycle generation. The next four columns, two each, report the corresponding statistics for the Benders decomposition based algorithms with algorithm $F$ and algorithm $I$ for cycle generation, respectively. The last three columns report the gap corresponding to the relative difference between the solution value of the $GI$ and the $BI$ algorithm, the $CI$ and the $BI$ algorithm and the $BF$ and the $BI$ algorithm, respectively. Initial cuts described in Section 2.4.2 are used in both of the Benders decomposition based algorithms. Also, columns with reduced cost less than -1,000,000 are removed after every 10 iterations, during the column generation phase in algorithm $C$ and while solving the master problem in Phase I of algorithm $B$. As discussed at the end of Section 2.3.3.2, care must be taken in setting the stopping conditions for the mixed integer program $BMP$ in Phase II of the Benders decomposition based algorithm.

In our computational experiments, stopping the MIP when a 1% optimality gap for small instances (6-10 ports) and a 3-5% gap for large instances (15-20 ports) is reached provided a good balance of computational time and solution quality. These parameters were set after initial computational experiments. Specifically, for the 6 port instances when the optimality gap is reduced from 1% to 0.1%, the solution quality improves by only $\sim 0.04\%$ whereas the time taken to solve the integer program increases by $\sim 55\%$. Hence we believe that heuristically solving the MIP's did not have a significant effect in prematurely terminating the Benders algorithm if the optimality gap was chosen properly. Also, in our computations when we use the above optimality gaps as stopping criterions we never ran into a situation where the upper bound obtained by the MIP was less than the Benders lower bound. For the $CI$, $BF$ and $BI$ algorithms, column # cycles reports the number of cycles in the integer program. Note that in these algorithms a larger number of cycles are generated during column generation, while solving the LP, but subsequently removed if they have high negative reduced cost.

The results of our tests show that there is a very significant difference in the solution quality obtained by the greedy algorithm and the solution quality obtained by the other two algorithms. Though the greedy heuristic is fast, it works with a very small set of cycles and picks each cycle that it generates without any further considerations. The pure column generation based algorithm yields solution qualities comparable to the Benders decomposition based algorithm with iterative cycle generation however it incurs a longer computational time and this difference increases as the problem size increases. Though the number of cycles passed on to the integer program in the pure column generation is not very high as compared to the number of cycles at the end of Phase I in algorithm $BI$, the amount of time taken is much higher. This can be contributed to the fact that as the problem size (number of ports, ships and demand triplets) increases the number of variables as well as the

**Table 1:** Comparison between different algorithms.

| Test Class | GI | | CI | | BF | | BI | | %GI -BI | %CI -BI | %BF -BI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #cycles | time | #cycles | time | #cycles | time | #cycles | time | | | |
| P6S18D6 | 2 | 0.03 | 62 | 0.95 | 234 | 0.68 | 49 | 0.16 | 49.27 | 0.01 | 3.92 |
| P6S18D9 | 3 | 0.05 | 70 | 2.10 | 272 | 2.59 | 64 | 0.30 | 57.84 | 0.60 | 2.90 |
| P6S30D6 | 3 | 0.09 | 181 | 7.26 | 215 | 1.30 | 96 | 0.33 | 37.28 | 1.10 | 2.81 |
| P6S30D9 | 4 | 0.13 | 239 | 34.42 | 366 | 1.73 | 120 | 0.57 | 33.01 | -0.21 | 1.99 |
| P10S30D18 | 5 | 0.76 | 312 | 760.20 | 1926 | 67.97 | 213 | 15.41 | 60.29 | -0.28 | 3.50 |
| P10S30D27 | 5 | 1.65 | 358 | 1312.00 | 2355 | 160.83 | 292 | 30.56 | 67.00 | 2.21 | 3.21 |
| P10S50D18 | 7 | 1.78 | 607 | 2077.10 | 3102 | 583.82 | 371 | 61.34 | 47.39 | 2.10 | 5.25 |
| P10S50D27 | 8 | 2.64 | 790 | 2910.00 | 4587 | 1318.68 | 578 | 116.35 | 45.01 | 0.02 | 5.65 |

number of constraints increases in the column generation based algorithm. However, in the Benders decomposition based algorithm the effect of increase in problem size is distributed between the master problem and the subproblem.

Though the $BI$ algorithm outperforms the $BF$ algorithm uniformly, the difference between the solution quality obtained by these algorithms is less than 6%. However, the time taken in the $BF$ algorithm is 4-5 times higher than the time taken by the $BI$ algorithm. This can be attributed to the fact that, in algorithm $BF$, many infeasible cycles are generated by solving the circulation problem and decomposing its flow in the first phase of the Benders decomposition based algorithm. For a 6(10) port problem $BF$ generated about 65%(60%) infeasible cycles in Phase I. Though more cycles are submitted at the end of Phase I by algorithm $BF$, the branch-and-bound takes far less time as compared to the corresponding branch-and-bound in algorithm $CI$ since most of the cycles generated by algorithm $BF$ are infeasible for the integer program and are removed at the start of the branch-and-bound. More over, in the $CI$ algorithm most of the time is spent in solving the LP relaxation via column generation.

Table 6 reports results for test cases with up to 10 ports because the pure column generation based algorithm and the flow decomposition based cycle generation algorithm become computationally very expensive thus making $CI$ and $BF$ ineffective. Also, the solution quality of the greedy algorithm decreases further as compared to the Benders decomposition based algorithm. Table 6 establishes the superiority of the solution, in terms of both CPU time and revenue generated, obtained by the two phase Benders decomposition based algorithm with iterative cycle generation. Thus, we used this algorithm to perform all further experiments.

**Table 2:** Analysis of the Benders decomposition based algorithm.

| Test Class | iters | Phase I | | | Phase II | %gap |
|---|---|---|---|---|---|---|
| | | sub-problem | master | cycle-gen | | |
| P6S18D6 | 13 | 0.02 | 0.11 | 0.09 | 0.01 | 10.24 |
| P6S18D9 | 16 | 0.10 | 0.19 | 0.14 | 0.03 | 12.10 |
| P6S30D6 | 20 | 0.06 | 0.23 | 0.17 | 0.03 | 2.30 |
| P6S30D9 | 27 | 0.16 | 0.36 | 0.27 | 0.05 | 3.32 |
| P10S30D18 | 47 | 7.24 | 6.13 | 5.17 | 1.99 | 8.55 |
| P10S30D27 | 56 | 17.02 | 7.65 | 3.63 | 6.54 | 9.80 |
| P10S50D18 | 75 | 23.87 | 20.64 | 16.09 | 19.65 | 1.91 |
| P10S50D27 | 95 | 52.69 | 31.20 | 18.37 | 35.55 | 3.24 |
| P15S45D42 | 130 | 105.86 | 69.46 | 52.27 | 35.25 | 8.63 |
| P15S45D63 | 175 | 141.60 | 110.69 | 72.00 | 33.80 | 8.53 |
| P15S75D42 | 181 | 172.25 | 152.49 | 118.80 | 167.72 | 5.30 |
| P15S75D63 | 200 | 254.26 | 212.56 | 156.08 | 174.78 | 5.92 |
| P20S60D76 | 200 | 1165.87 | 73.12 | 39.92 | 42.07 | 12.70 |
| P20S60D114 | 200 | 1750.63 | 113.18 | 47.38 | 173.37 | 7.51 |
| P20S100D76 | 200 | 2507.51 | 164.61 | 72.81 | 262.45 | 5.05 |
| P20S100D114 | 200 | 3784.38 | 380.11 | 149.65 | 478.01 | 7.21 |

### 2.5.3 Analysis of the Benders Decomposition Based Algorithm

Our next set of experiments perform a deeper analysis of the Benders decomposition based algorithm and are presented in Table 7. In these experiments we used initial cuts and removed columns with large negative reduced costs after every 10 iterations in the first phase of the Benders decomposition based algorithm. The second column in Table 7 represents the number of iterations in the first phase of the algorithm. The third, fourth and fifth columns present a breakdown of the total time taken in various processes while solving the $LPBMP$. The next column represents the additional time taken to obtain an integer solution. The last column reports the gap corresponding to the relative difference between the upper bound, obtained by the $CI$ algorithm, and the integer solution value obtained by the $BI$ algorithm. To keep computational time under control, in Phase II, only 2-3 iterations of the Benders algorithm were performed.

Table 7 suggests that as the number of demand triplets increase the time taken

in solving the sub-problem increases. This is mainly because every demand triplet is considered a different commodity thus as the number of demand triplets increase the complexity of the multi-commodity flow problem or the subproblem increases (in the number of variables and constraints) significantly. Note that an increase in the number of demand triplets results in an increase in the time taken to solve the master problem also. This is because of the increased possibilities with regard to the cycles that can be generated. The overall time increases as we increase the number of ports, the number of ships or the number of demand triplets.

For the same number of ports, as the number of ships increases the integrality gap reduces significantly. This suggests that the set of cycles generated in the first phase are good for the second phase also and given sufficient number of ships the gap can be reduced further. For small test cases with 6 ports, we observed that the integer solution obtained by our algorithm is indeed close to the optimal solution in many cases and that LP based upper bound is not very tight. It is easily seen that the integrality gap can be very bad. Consider a two port, one ship instance such that the sailing time between ports is one week. An LP solution will assign half a ship to each edge whereas an integer solution will yield zero revenue resulting in a 100% integrality gap. However, given a sufficient number of ships such extremely pathological cases are highly unlikely to occur.

Our next set of experiments studies the effect of using the refinements described in Section 2.4.2 and Section 2.4.3. Using the two phase approach we solve each instance first without the initial set of cuts, then without removing any column at intermediate steps and finally by incorporating the initial cuts and removing columns at intermediate steps to keep only a subset of columns. Parameters are chosen so that the solution quality is not affected by these refinements however the computational time is reduced significantly. Table 8 reports cycles generated, iterations performed and the time taken for each of these cases. The total CPU time taken to find an

**Table 3:** Effect of algorithmic refinements.

| Test Class | No Cuts + All Cols | | | Cuts + All Cols | | | Cuts + Remove Cols | | |
|---|---|---|---|---|---|---|---|---|---|
| | #cycles | iters | time | #cycles | iters | time | #cycles | iters | time |
| P6S18D6 | 62 | 15 | 0.20 | 51 | 13 | 0.17 | 49 | 13 | 0.16 |
| P6S18D9 | 93 | 15 | 0.34 | 87 | 15 | 0.33 | 64 | 16 | 0.30 |
| P6S30D6 | 194 | 22 | 0.66 | 105 | 20 | 0.44 | 96 | 20 | 0.33 |
| P6S30D9 | 240 | 30 | 1.03 | 131 | 27 | 0.88 | 120 | 27 | 0.57 |
| P10S30D18 | 494 | 52 | 18.82 | 464 | 45 | 18.68 | 213 | 47 | 15.41 |
| P10S30D27 | 673 | 60 | 64.50 | 603 | 55 | 50.45 | 292 | 56 | 30.56 |
| P10S50D18 | 882 | 79 | 271.03 | 790 | 71 | 168.82 | 371 | 75 | 61.34 |
| P10S50D27 | 1102 | 99 | 748.23 | 889 | 91 | 364.81 | 578 | 95 | 116.35 |

integer solution is also reported.

Table 8 reports results for networks with up to 10 ports because the time taken in both phases of the Benders decomposition based algorithm becomes prohibitively high, for networks with more than 10 ports, if we remove the initial cuts or do not remove cycles with large negative reduced cost. Note that removing columns with negative reduced cost less than -1,000,000 does not reduce the number of cycles significantly for 6 port instances since not many cycles for such a small network have a large negative reduced cost. However the same refinement reduces the number of cycles for 10 port instances to approximately half the size suggesting that this refinement must be tuned according to the problem size to properly control the number of columns in the linear program.

Table 8 suggests that the CPU time as well as the number of iterations, in the first phase of the Benders decomposition based algorithm, reduce by introducing the initial cuts. However, a more significant reduction in time is achieved by removing columns with large negative reduced cost. Removing very negative reduced cost cycles does not affect the time taken in Phase I very much, but the number of columns that the integer program works with in Phase II are reduced considerably and thus the time taken in the second phase of the Benders decomposition based algorithm reduces significantly.

Finally, we study the effect of having only one ship type, in the fleet, on the solution quality. Table 4 reports results for a fleet of identical ships with 4000 TEU capacity. For each test class, we report the CPU time taken in Phase I and Phase II of the Benders decomposition based algorithm with iterative search for cycle generation, the total number of cycles generated and the optimality gap. In this case also, 2-3 iterations of the Benders algorithm were performed in Phase II.

**Table 4:** Effect of identical ships in the fleet.

| Test Class | Phase I | | | Phase II | #cycles | %gap |
|---|---|---|---|---|---|---|
| | sub-problem | master | cycle-gen | | | |
| P6S18D6 | 0.02 | 0.09 | 0.07 | 0.00 | 35 | 1.43 |
| P6S18D9 | 0.04 | 0.17 | 0.14 | 0.01 | 42 | 2.14 |
| P6S30D6 | 0.03 | 0.11 | 0.09 | 0.00 | 60 | 0.16 |
| P6S30D9 | 0.06 | 0.20 | 0.18 | 0.04 | 72 | 2.01 |
| P10S30D18 | 3.92 | 2.97 | 2.41 | 0.30 | 190 | 2.25 |
| P10S30D27 | 5.06 | 2.16 | 1.82 | 0.62 | 202 | 2.23 |
| P10S50D18 | 8.32 | 6.98 | 5.25 | 2.02 | 243 | 1.45 |
| P10S50D27 | 13.38 | 8.36 | 6.82 | 2.54 | 275 | 1.74 |
| P15S45D42 | 51.72 | 17.40 | 15.35 | 3.72 | 398 | 2.53 |
| P15S45D63 | 97.05 | 25.59 | 21.97 | 4.97 | 520 | 2.01 |
| P15S75D42 | 171.12 | 52.77 | 37.77 | 5.20 | 583 | 1.93 |
| P15S75D63 | 209.07 | 87.28 | 52.78 | 6.83 | 647 | 1.56 |
| P20S60D76 | 1023.53 | 106.96 | 91.60 | 12.83 | 450 | 1.32 |
| P20S60D114 | 1869.77 | 193.79 | 117.76 | 13.50 | 791 | 1.16 |
| P20S100D76 | 1825.67 | 181.85 | 144.67 | 14.82 | 957 | 1.91 |
| P20S100D114 | 2923.28 | 189.13 | 141.51 | 16.50 | 980 | 2.01 |

Table 4 suggests that if all the ships are identical the optimality gap reduces even further in all the test classes. Since all ships are identical, in Phase II it becomes easier to operate a service route using ships of similar kind to maintain the weekly frequency. Comparing Table 7 to Table 4 suggests that the overall time taken also reduces. The time taken in the cycle generation process reduces significantly as now the cycle generation needs to be solved only for one ship type at every iteration. Thus the time taken in the master problem decreases. Also note that a fewer number of cycles are generated and thus the time taken in Phase II reduces significantly. As a

result the overall solution time is reduced.

### 2.5.4 Analysis of the Solution

In this section, we take a closer look at the solution generated by the Benders decomposition based algorithm and its implications. Also, we perform preliminary experiments to study the effect of transshipment cost on cargo routing.

The second column in Table 5 reports the number of cycles or service routes picked in the final solution. The number of service routes increases as the number of ships and the number of ports increase. The next two columns in Table 5 report the average percentage utilization of capacity on the edges of the network and the percentage of the cargo that is transshipped. These results are for the case when we do not consider transshipment cost i.e. the cost of transshipment is 0. Utilization of capacity on an edge is calculated by dividing the total flow on that edge by the total capacity of the edge. Recall that the capacity of an edge is defined by the number of ships (and their capacities) that utilize the given edge. Across our problem instances, our algorithm consistently reports high average percentage utilization, 70-90%, of capacity. Note that higher the number of service routes, higher is the number of possibilities for cargo routes. As a result the percentage of the cargo transshipped to the total cargo shipped increases as the problem size increases. This trend is observed in our computational study also as the amount of transshipped cargo increases from $\sim 19\%$ for a 6 port problem to $\sim 30\%$ for a 10 port problem.

Next, we perform preliminary experiments to study the effect of transshipment cost on cargo routing. Depending on the set of chosen service routes we construct a new network. In the new network, at every port where two or more cycles meet a new node is constructed for every cycle. The new nodes are connected to the original port node via edges. These edges act as loading/unloading edges and have corresponding costs associated with them. For example, for the network represented in Figure 4

**Table 5:** Analysis of the obtained solutions.

| Test Class | # picked -cycles | Trans_cost =0 | | Trans_cost =20 | | Trans_cost =100 | | Trans_cost =1000 | |
|---|---|---|---|---|---|---|---|---|---|
| | | % utili -zation | % trans shipped | % diff demand | % trans shipped | % diff demand | % trans shipped | % diff demand | % trans shipped |
| P6S18D6 | 3 | 0.58 | 17.28 | 0.00 | 12.84 | 0 | 12.84 | 30.99 | 15.17 |
| P6S18D9 | 3 | 0.82 | 21.26 | 0.00 | 17.12 | 0.65 | 16.86 | 28.96 | 6.41 |
| P6S30D6 | 4 | 0.71 | 20.03 | 0.00 | 13.77 | 0.27 | 13.48 | 53.54 | 0.01 |
| P6S30D9 | 5 | 0.79 | 17.33 | 0.00 | 14.47 | 1.59 | 13.48 | 33.79 | 6.58 |
| P10S30D18 | 6 | 0.83 | 22.30 | 0.00 | 15.63 | 0.66 | 14.99 | 23.00 | 3.71 |
| P10S30D27 | 6 | 0.86 | 28.07 | 0.00 | 23.08 | 0 | 23.08 | 13.94 | 11.96 |
| P10S50D18 | 7 | 0.88 | 34.09 | 0.25 | 29.46 | 1.08 | 28.81 | 31.79 | 14.42 |
| P10S50D27 | 8 | 0.91 | 32.53 | 0.26 | 27.84 | 0.74 | 27.44 | 30.44 | 11.41 |

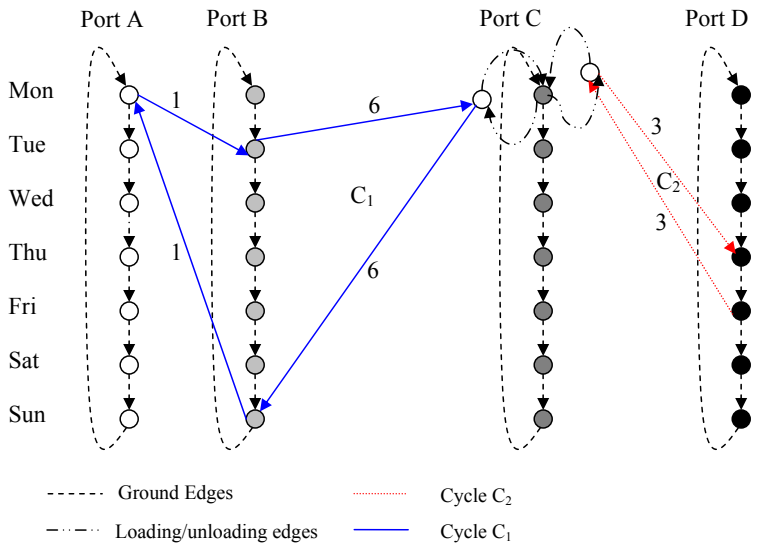


**Figure 7:** New network to study the effects of transshipments

the new network is given by Figure 7. At port $p$, $c_p^u$ and $c_p^l$ denote the unloading and loading cost respectively and the transshipment cost is given by $c_p^u + c_p^l$. Thus a transshipment occurs when at an intermediate port cargo travels on an unloading and then a loading edge. In Figure 7, a cargo that is routed from port $B$ to port $D$ is transshipped at port $C$ and it uses the unloading edge from cycle $C_1$ to port $C$ and the loading edge from port $C$ to cycle $C_2$.

To perform the experiment, we construct the new network for the cycles selected at the end of the second phase of the Benders based algorithm. The cargo routing problem is solved for both the, new and the original, networks. The effect of the transshipment costs on the cargo routing decisions is studied by observing the percentage difference between the demand satisfied in the original network (in the absence of transshipment costs) and the new network (in the presence of transshipment costs). Also we compute the percentage of cargo transshipped to the total cargo shipped. These two statistics are reported in Table 5 for three different scenarios: transshipment cost = 20 units/per unit of cargo, transshipment cost = 100 units/per unit of cargo and transshipment cost = 1000 units/per unit of cargo. Recall that the holding cost at ports is chosen randomly from $[1, 10]$ and the revenue generated by satisfying

demand is chosen to be proportional to the distance (proportionality constant being chosen randomly from $[100, 200]$) between the origin and destination ports. Note that as the distance between ports is chosen from $[2, 42]$ days, the revenue generated is chosen from $\sim [200, 8000]$. Thus the first scenario represents the case when the transshipment cost is low and is comparable to the holding cost at a port. The third scenario represents the case when the transshipment cost is very high and is comparable to the revenue generated by satisfying demand. Such high transshipment costs are highly unlikely however we discuss this scenario to present an extreme case.

Our computations yield that when the cost of transshipment is of the order of the holding cost at a port or low as compared to the revenue generated by satisfying demand, the routing decision in both networks are similar. However, as the transshipment cost increases the routing decisions change. Specifically, as the transshipment cost increases from 20 to 1000 units the percentage change in the amount of demand satisfied increases from 0% to $\sim$ 36%. We note that as the transshipments become more and more expensive the percentage of the cargo transshipped to the total cargo shipped decreases. An anomaly occurs in the first row last column of Table 5 as the percentage of transshipped cargo increases from 12.84% to 15.17% when the transshipment cost increases from 100 units to 1000 units. This occurs because as the transshipment cost increases not only does the transshipments decrease but also the demand that is satisfied decreases in many cases since the routing options get limited. Thus for the last column the numerator as well as the denominator decreases. For instances in the class $P6S186$ the denominator decreases faster than the numerator because for this class of instances we have very few demand pairs (few things to route) and on average very few selected cycles (very few alternative routing options).

## 2.6 Concluding Remarks and Future Research

In this chapter we presented a new mathematical model for the simultaneous ship scheduling and containerized cargo routing problem for liner shipping. The proposed model captures the important weekly frequency constraint faced by the carriers and allows them to take advantage of transshipping cargo. The structure of the model makes it well suited for decomposition, leading to efficient algorithms. Effective service routes for ships are generated selectively in a column generation setting using an iterative search algorithm. Finally, the proposed solution approach is tested on various test classes. Considering the preliminary results obtained, we believe that the suggested solution approach has the potential to help the planners in developing better routes for a fleet of up to 100 ships. The planners can also add their predetermined service routes to the model as a set of initial cycles and thus be a part of the solution process to obtain a solution which is a "user's solution" rather than a "computer's solution". Our results indicate high percentage utilization of ships' capacities and a significant number of transshipments in the final solution.

Our aim in this chapter is to provide a basic framework for simultaneous ship scheduling and cargo routing. The model and the solution strategy presented here can be enhanced in different ways. Next, we present some directions for future research. The model presented in this chapter allows for transshipping the cargo from one ship to another. At the end of Section 2.5.4 we presented an approach to account for transshipment costs during cargo routing. However, the model does not take in to account the transshipment costs while designing the service routes. Further research is required to extend or modify the model to include transshipment costs. This aspect is expected to increase the complexity of the model and the solution procedures significantly.

In our model we allow only one ship type to maintain weekly frequency on a service route. This provides same capacity in the network every week and is useful

when a carrier faces same demand each week. However, it is possible that the demand structure is not the same each week. Further research is required to allow for multiple ship types on a service route. Changes in demand from week to week can be incorporated easily by expanding the planning horizon, however incorporating cycles with multiple ship types will require changes in the model and the cycle generation scheme.

In the pure column generation algorithm and the Benders decomposition based algorithm, no new columns are generated when solving the integer program. New columns can be generated by solving the integer program in a branch-and-price (rather than the branch and bound used here ) framework. Branch-and-price is expected to improve the solution quality. However, there are many important and challenging issues that are required to be resolved for developing a successful branch-and-price algorithm. Specifically, a good branching rule needs to be devised. Standard branching on the cycle or the $x_C$ variables creates a problem along the branch where a variable has been set to zero. $x_C = 0$ means that cycle $C$ needs to be excluded. However, it is possible that the next time the pricing problem is solved to generate a profitable cycle in this branch, the optimal solution is precisely the cycle $C$. Thus the second best cycle must be considered. More over, at depth $l$ in the branch and price tree it might be necessary to construct the $lth$ best cycle. Note that a successful branch-and-price algorithm requires a pricing problem that can be solved very efficiently, as it will be invoked many times. Explicitly excluding the specified cycles from the pricing problem is computationally expensive. Even if a pool of cycles is generated at every column generation step, one needs to keep track of all the cycles that need to be excluded. Since commercial softwares such as CPLEX cannot handle the branch and price framework, managing the search tree efficiently poses many implementation challenges, such as deciding which nodes to branch on and which search technique e.g. breadth first search, depth first search, best bound, etc to use.

# CHAPTER III

# ALLIANCE FORMATION AMONG SEA CARRIERS

## *3.1 Introduction*

In order to position themselves better against their competitors carriers rely on good customer service (shorter transit times, higher frequency of service) and competitive prices. However, some of the recent trends in the industry such as increasing customer expectations, shrinking profit margins, new entrants (for example brand name package delivery providers such as DHL and UPS that are synonymous with reliability and speed) and new markets [45] are leading to cutthroat price competition among various carriers. These market and customer pressures are driving carriers to adopt solutions outside of their traditional business practices and identify competitors with most synergies to increase profit margins and meet increasing customer expectations. Though carriers collaborate and form alliances on many trade routes they are competitors with selfish interests. Sustainable collaborations require mechanisms to govern membership rules and allocate costs and benefits in a fair way. As observed by Jain and Vazirani [36] such problems appear in a wide variety of seemingly unrelated fields such as internet routing, auctions, telecommunications and transportation and have the following two properties in common:

1. In all of these problems a number of players/participants interact with varying degree of collaboration and self motives.

2. The underlying computational problem is NP hard.

The mathematical tools and insights most appropriate to understand and analyze these problems are obtained by uniting concepts of mathematical economics and game

theory with that of algorithm design. The linear programming machinery, especially ideas from duality theory such as primal dual methods, are heavily used in literature to develop good algorithms for solving these problems [30], [36].

In this chapter and Chapter 5 we study strategic alliance formation among two or more carriers in containerized liner shipping. Sea carriers collaborate to form operational, pricing and logistical alliances. We study alliances, among sea carriers, that are formed by pooling, exchanging and integrating ships in their fleet.

In this chapter, we first study the reasons that motivate carriers to collaborate and form alliances. In Section 3.3, we review some relevant game theoretic concepts. Carriers form alliances by pooling their ships and integrating their networks. The maximum revenue that an alliance can generate can be obtained by replacing individual carriers with one large carrier, with a fleet equal to the combined fleet of different individual carriers and a demand structure equal to the combined demand of all carriers, and solving the corresponding optimization problem for a single carrier as formulated in Chapter 2. In Section 3.4, we present a small example to study alliances among liner carriers. In Section 3.5, for a special case of the problem, we provide a fair allocation of the total revenue generated by the alliance among its members. Allocation mechanisms for the general case are presented in Chapter 5.

## *3.2 Why Collaborate?*

Traditionally, companies have focused on their own resources and ability to perform effectively and efficiently. However, recently in logistics and supply chain management, companies that in the past worked independently of each other, are working in close liaison. The system wide collaboration perspective provides opportunities for increased profitability that are impossible to achieve through internal focus only. Vendor managed inventory (VMI) and shipper collaboration (for example the success of collaborative logistics networks run by Nistevo and Transplace) are two of

the most successful applications of collaboration in logistics to date. Globalization, containerization, deregulation and easy integration of logistics networks of different carriers due to technological advancements have led to similar trends in the sea cargo industry.

Collaboration among sea carriers is not new. Carriers have used *conferences*, as a means for curbing competition and controlling tariff rates in the market, for over a century. The first conference was formed in 1875 on a route between the United Kingdom and Calcutta, India. More recently, carriers are forming strategic alliances that allow them to share capacity on a ship and to share slots at the ports. Since 1990, when Sea-Land and Maersk introduced the alliance system and began sharing vessels in the Atlantic and Pacific oceans, strategic alliances have become increasingly common. The industry is becoming more and more consolidated (for example Maersk's bid for P&O Nedlloyd in 2005) and smaller alliances are collaborating to form even bigger alliances (for example Grand Alliance and The New World Alliance laid down foundation for cooperating in 2006). Figure 8 depicts some of the trends of consolidation in 2007 as compared to those in 1995. More over, the shipping industry in many nations including the United States has enjoyed anti-trust impunity because of the widely accepted fact that this industry is highly capital intensive and collaboration among carriers helps provide regular service between ports. According to [45], in near future the top 10 carriers will control about 80% of the market with the next 20 carriers controlling about 15% of the market. We now list some of the motivating factors for alliance formation among sea carriers. These have been studied by [67] in detail.

1. In the last couple of decades, many factors have led to the consolidation of manufacturing sector thus leading to bigger demands. This consolidation works in favor of the shippers who can now control a bigger share of the market. However this leads to squeezed profit margins for the sea cargo carriers. Sea

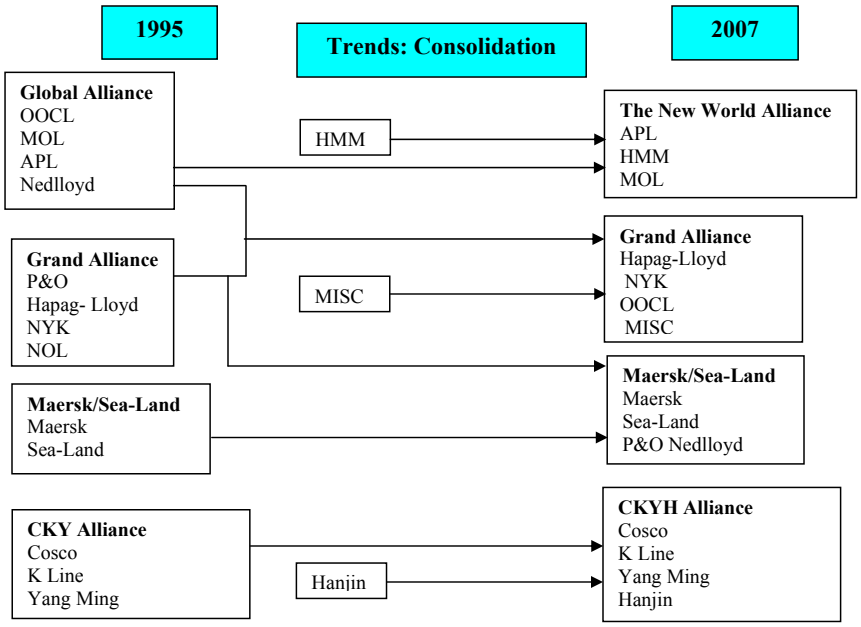


**Figure 8:** Trend of consolidation

cargo carriers have counter acted by forming many alliances among themselves to help them fix prices so that the shippers have less bargaining power.

2. Liner shipping is a capital intensive industry with infra-structural set up as its backbone. Carriers need to invest heavily in assets such as ships (owning a ship involves millions of US dollars and the cost of idling a 2,000 TEU ship is $20,000-$25,000 per day), containers and land based facilities such as marine terminals etc. Carriers collaborate and form alliances to reduce and share capital costs. Huge costs are involved in maintenance and operation of ships as well. In extreme cases of collaboration carriers form alliances with carriers who do not own any ships or "NVO"s(Non-vessel operator). NVOs compensate ship owners for using capacity on their ships and for ship owning carriers they act like shippers with big demands. Similarly, a carrier who has invested heavily in land based infra structure and marine terminals can improve the utilization of these facilities by getting its partner's cargo volume to assist in spreading the

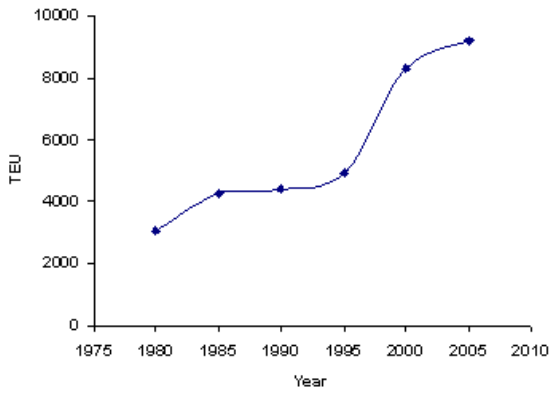


**Figure 9:** Increase in size of container ships. Source: Wikipedia

capital investment costs. Advent of containerization of cargo in 1990s demanded new investments and many alliances at that time were formed to share this cost.

3. Ships are becoming larger and larger, as depicted by Figure 9, thus giving rise to chronic over capacity. A recent increase in literature e.g. [35] on the viability of larger ships is also a pointer towards the increasing use of bigger ships. The construction cost, the cost of operation (e.g. fuel, equipment, port visit cost e.t.c) and the cost of maintenance (e.g. crew cost, repair cost e.t.c.) on a ship does not increase in proportion to the size of the ship. For example, in 1992 when ship size increased from 800 TEUs to 2,500 TEU's (or 212%), the construction cost increased by only 160%. This motivates construction of bigger and bigger ships. However, the capacity on the ship is perishable. Once the ship leaves the port the capacity becomes unusable until it reaches a loading port again. Alliances provide carriers with opportunities to deploy bigger ships, thus achieving reduction in cost via economies of scale and higher utilization of space, by catering to the demand of multiple carriers. For example, two carriers with 2,000 TEUs demand each week can deploy a single 4000 TEU ship rather than two 2,000 TEUs ships and thus achieve economies of scale while still satisfying their individual demands.

4. Sea cargo industry is marked with low product differentiation. Almost all carriers have same facilities and ships with similar speeds. Containerization of cargo further reduces any differences. The low differences help carriers to form alliances easily.

5. Good frequency of service is essential for a carrier to achieve higher market share. The "Just-in-time" inventory management system deployed by many businesses demands timely and frequent transportation service. Thus, most carriers have at least one departure each week from each port visited on a service route. As observed in the previous chapter, this requires that the number of ships that operate on a cycle be at least equal to the number of weeks that it takes to complete a cycle. However some cycles such as the Asia- North America cycle take up to eight weeks to complete. This means many ships are required to maintain good frequency on these routes. However if two carriers with 2,000 TEU demand from Asia to North America run this eight week service route in collaboration then each carrier would need to contribute only four ships of 4000 TEUs and still have access to a weekly 2000 TEU capacity, with a proportional allocation of space on the ships. This can also be viewed as a way of breaking the discreteness of the ships. In this example we can say that a carrier is running half a ship each week. Note that if a carriers has less than eight ships then he cannot offer any service on this very profitable trade route. More over, in the above example, if both carriers have eight ships each of 4000 TEU to offer for the service route then they can increase frequency on the route and potentially grab a higher market share. For example according to 2005 data [47], the Grand Alliance, which is the largest integrated alliance, offers a total capacity of 640,000 TEUs with 112 vessels dedicated to the three main East-West trade lanes, thus maintaining its strong hold on these trade lanes by offering faster transit times and high frequency service. Pooling ships together

allows carriers to group ships with similar characteristics together, independent of ownerships, to offer same services every week on jointly operated routes, by deploying compatible ships on a service route.

6. Alliances help carriers to explore new markets and enhance their global reach. Transshipments play a vital role in enhancing global markets. For example in Figure 10 let carrier $A$ and carrier $B$ operate cycle $C_1$ and $C_2$ respectively. To obtain benefit from an emerging market from port $P_1$ to port $P_3$ they can form an alliance by offering space on their respective cycles and by transshipping at port $P_2$. Thus both of them have expanded markets without deploying any additional ships. This is well suited for shippers as well as they can avoid dealing with multiple carriers.

   Alliances also help carriers redistribute their excess capacity and de-emphasize or emphasize their operations in some areas. These needs often arise as a result of change in corporate mission and internal decision regarding which trades to participate in, exit, trim or expand. For example, in 2005 MISC, MOL, NYK, OOCL and PIL started two collaborative routes to cover China, Singapore and New Zealand. The strategic alliance allowed them to extend their services to new destinations thus enhancing their role in those regions and generating new opportunities for mutual growth. Similarly in 1993, Hapag Lloyd formed an alliance with NOL and NYK, who offered Far East -North America service. This alliance forced Hapag Lloyd to reduce its own weekly capacity on its trans Atlantic route, to offer space to the alliance. However, this provided Hapag Lloyd with an opportunity to access the trans Pacific trade and thus a chance to redistribute its trans Atlantic capacity to the trans Pacific.

7. In spite of globalization, carriers tend to dominate their home markets and this trend is expected to remain more or less the same. There are many reasons

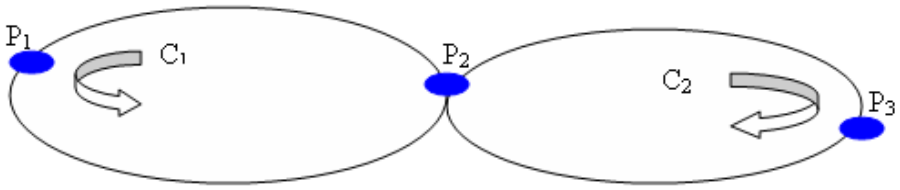


**Figure 10:** Transshipment at port $P_2$

such as a better understanding of the political and geographical issues and human instincts for this behavior. However this leads to huge imbalance in trade in many cases. With an industry concerned with filling ships, alliances help carriers to reduce the imbalance in trade and achieve better utilization of ship's capacity. For example two carriers, one with a high demand from North America to Asia and another with a high demand from Asia to North America can find perfect synergies for alliance formation.

Finally, we conclude that though the notion of cooperation runs contrary to the concept of perfect competition, carriers form alliances to realize economies of scale, extend customer base, increase asset utilization, regulate traffic and fix prices while providing customers with more frequent sailings and faster transit times. Decisions regarding the alliance routes have to be taken together by the alliance members. However, it involves management costs etc and all partners have their own interests in mind. Thus, to form successful alliances, carriers need to identify suitable partners with most synergies. In many cases collaborators have parted ways because they could not align their interests properly. For example, Ben Lines and P&O dissolved their alliance because of their similar market focus in the east Europe trade. Though alliances provide more opportunities to carriers, they make the network design complicated and give rise to issues of distribution of benefits and costs. We consider these issues next. But first we introduce some game theoretic definitions and concepts that are relevant to this and the following chapters.

## 3.3 Game Theory Basics

A *cooperative game* is defined by a set $N$ of players and a *characteristic function* $opt : 2^N \rightarrow \mathbb{R}$. $opt(S)$ maps a value to every subset/coalition $S \subset N$, interpreted as the total gain the members of $S$ can achieve by cooperating. We assume that $opt(\emptyset) = 0$. The set $N$ itself is referred to as the *grand coalition*. The central problem in cooperative game theory is how to allocate the total gain $opt(N)$ among the individual players $i \in N$ in a "fair" way. We denote an allocation/payoff vector by $x = \{x^1, \ldots, x^n\} \in \mathbb{R}$, where $x^i$ refers to the payoff made to player $i$.

One of the most prominent and widely accepted notions of fairness is the "core" of the game. An allocation of benefits is said to be in the core if the sum of the payoffs over all players is their maximum attainable profit (budget balance property) and no subset of players can collude and obtain a better payoff for its members (stability property). Mathematically, a payoff vector $x$ is said to be in the core if:

$$\sum_{i \in N} x^i = opt(N) \tag{28}$$

$$\sum_{i \in S} x^i \geq opt(S) \quad \forall S \subset N. \tag{29}$$

## 3.4 A First Look

As challenging as the optimization problem of network design for a set of carriers is, it is also important to identify the rules governing the distribution of benefits among the carriers. Assuming that a carrier is concerned only with a payoff at the end and is indifferent towards which demand is satisfied and which service routes are realized *the sea cargo network design game* can be defined as: Given a set of carriers $\mathcal{N} = \{1 \cdots n\}$ and specific demand set $\Theta_k$ and $N_k^a$ ships of type $a$ for each carrier $k$, carriers form alliances by consolidating their demand, $\Theta = \cup_k \Theta_k$, and pooling their ships, $N^a = \sum_k N_k^a$. Together they need to decide on a set of service routes for their ships and select a set of cargo to be shipped on the chosen routes, to maximize the

overall revenue generated by the alliance. Then they need to identify a "fair" payoff method to allocate the revenue generated by the alliance among its members. Note that in our study we ignore any costs, logistical and management, involved in forming such alliances.

We refer to the participating carriers as players and any non empty subset of $\mathcal{N}$(including $\mathcal{N}$ itself and all the one element subsets) as a coalition. For a subset $S$ of carriers we denote by $opt(S)$ the schedule obtained by solving the optimization problem, as described in Chapter 2, when the demand and fleet size is restricted to the carriers in the set $S$. By the characteristic function of the sea cargo network design game we mean a real-valued function $r(opt)$ defined on the subsets of $\mathcal{N}$, which assigns to each $S \subset \mathcal{N}$ the profit attainable by the schedule $opt(S)$. In other words, $r(opt(S))$ is the amount of payoff that the carriers in $S$ can obtain, whatever the remaining carriers may do. As observed in Chapter 2, determining $opt(S)$ itself is a hard problem. Also, we define a payoff allocation vector by $x = \{x_1, ..., x_n\}$, where $x_k$ is the payoff allocated to carrier $k$.

A payoff allocation in the core represents a very strong type of stability (where the grand alliance is not threatened by sub-coalitions) and provides a fair allocation of benefits among the members. However, frequently the core of a game is empty. Shapley [60] proved that if the characteristic function of a game is supermodular (i.e. $r(opt(S)) + r(opt(T)) \leq r(opt(S \cup T)) + r(opt(S \cap T))$) then the core is nonempty. We now present a sea cargo network design game instance for which the characteristic function is not supermodular.

Consider an instance with three ports with distances (in terms of the number of weeks that a ship takes to reach from one port to another) as depicted in Figure 11. Consider three carriers $A$ with two ships and $B$ and $C$ with one ship each. Carrier $A$ has demand from port $P_2$ to port $P_3$ and carrier $B$ and carrier $C$ have demands from port $P_1$ to port $P_2$. Assume all ships are identical with 1000 TEU capacity, all

demands are unlimited, unit amount of revenue can be generated by satisfying a unit of demand and there are no costs involved. Let us consider the set $S = \{A, B\}$ and $T = \{B, C\}$. Then, after maintaining a weekly frequency on operated cycles, clearly

$r(opt(S)) = \$2000$: 3 ships on cycle $P_1 - P_2 - P_3 - P_1$.

$r(opt(T)) = \$1000$: 2 ships on cycle $P_1 - P_2 - P_1$.

$r(opt(S \cup T)) = \$2000$: 3 ships on cycle $P_1 - P_2 - P_3 - P_1$.

$r(opt(S \cap T)) = \$0$ : Carrier B cannot operate on its own.

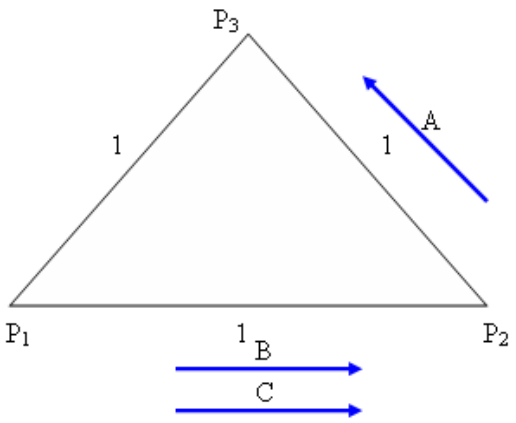


**Figure 11:** The sea cargo network design game is not super-modular

Since in the above instance, $r(opt(S \cup T)) + r(opt(S \cap T)) \not\geq r(opt(S)) + r(opt(T))$ we do not have supermodularity. Also, in the above example it is easy to see that

$$r(opt(\{A\})) = r(opt(\{B\})) = r(opt(\{C\})) = \$0$$
$$r(opt(\{A, B\})) = r(opt(\{A, C\})) = \$2000, r(opt(\{B, C\})) = \$1000$$
$$r(opt(\{A, B, C\})) = \$2000.$$

Thus for an allocation $\{x_A, x_B, x_C\}$ in the core, for the budget balance condition we must have

$$x_A + x_B + x_C = 2000 \tag{30}$$

and for the two carrier subset rationality constraints we must have

$$x_A + x_B \geq 2000 \tag{31}$$

$$x_A + x_C \geq 2000 \tag{32}$$

$$x_B + x_C \geq 1000 \tag{33}$$

$$\Rightarrow x_A + x_B + x_C \geq 2500. \tag{34}$$

Thus, the core in this example is empty. Next, we look at a subclass of the general sea cargo network design game where an allocation in the core can always be found.

## *3.5 An Allocation in the Core*

In this section we consider a subclass of the general sea cargo network design game, namely the instances for which the the integer program (1) - (7) in Chapter 2 and its linear programming relaxation have the same objective function value. For these instances an allocation in the core can always be found with the help of the linear programming machinery. We provide an algorithm, which utilizes the dual of the problem, for obtaining an allocation in the core.

The use of dual variables for determining a payoff allocation in the core can be traced back to the classic Bondareva-Shapley theorem [12] , [59]. It has been shown that core allocation can be obtained as a function of dual variables for problems such as the facility location game [30], in the special case when there is no integrality gap for the corresponding linear programming relaxation, and for problems such as the multicommodity flow game [42] in general.

Consider the LP relaxation of the integer linear program (1)-(7) from Chapter 2 where $\Theta = \cup_k \Theta_k$ and $N^a = \sum_k N^a_k$. As in Chapter 2, let $\pi = \{\pi_v : \pi_v \geq 0 \ \ \forall v \in G\}$, $\lambda = \{\lambda_e : \lambda_e \geq 0 \quad \forall e \in E_v\}$, $\omega = \{\omega^{(o,d,i)} : \omega^{(o,d,i)} \geq 0 \quad \forall (o,d,i) \in \Theta\}$ and $\sigma = \{\sigma_a : \sigma_a \geq 0 \ \ \forall a \in \mathcal{A}\}$ be the dual variables associated with constraints (2), (3), (4) and (5) respectively. The dual of the LP relaxation can be written as:

$$\min \sum_{(o,d,i)\in\Theta} \omega^{(o,d,i)} D^{(o,d,i)} + \sum_{a\in\mathcal{A}} \sigma_a N^a \tag{35}$$

such that

$$\begin{aligned}
\pi^{(o,d,i)}_{head(e)} - \pi^{(o,d,i)}_{tail(e)} + \lambda_e &\geq -c^c_e \quad \forall e \in E - E_f \ , \ \forall (o,d,i) \in \Theta && (36)\\
\pi^{(o,d,i)}_{head(e)} - \pi^{(o,d,i)}_{tail(e)} + \omega^{(o,d,i)} &\geq R^{(o,d,i)} - c^c_e \quad \forall e \in E_f \ , \ \forall (o,d,i) \in \Theta && (37)\\
-T^a \sum_{e\in C} \lambda_e + L_C \sigma_a &\geq -Cost_C \quad \forall a \in \mathcal{A}, \quad \forall \mathcal{C} && (38)\\
\pi, \sigma, \lambda, \omega &\geq 0. && (39)
\end{aligned}$$

Let $(\pi^*, \lambda^*, \omega^*, \sigma^*)$ be an optimal solution to the above linear program.

**Lemma 2.** *If there is no integrality gap for the LP relaxation of the sea cargo network design game, the payoff vector $x = \{x_1, ..., x_n\}$ such that*

$$x_k = \sum_{(o,d,i)\in\Theta_k} \omega^{(o,d,i)^*} D^{(o,d,i)} + \sum_{a\in\mathcal{A}} \sigma^*_a N^a_k \tag{40}$$

*provides an allocation in the core.*

*Proof.* We now prove that the allocation in (40) satisfies both conditions required for the definition of core.

1. Budget balance

$$\begin{aligned}
\sum_{k\in\mathcal{N}} x_k &= \sum_{k\in\mathcal{N}} \left( \sum_{(o,d,i)\in\Theta_k} \omega^{(o,d,i)^*} D^{(o,d,i)} + \sum_{a\in\mathcal{A}} \sigma^*_a N^a_k \right)\\
&= \sum_{(o,d,i)\in\Theta} \omega^{(o,d,i)^*} D^{(o,d,i)} + \sum_{a\in\mathcal{A}} \sigma^*_a N^a\\
&= r(opt(\mathcal{N})).
\end{aligned}$$

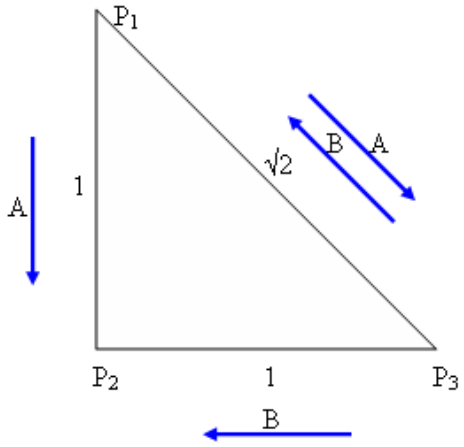


**Figure 12:** An allocation in the core may exist even when the LP relaxation has an integrality gap

2. Stability

   Let $S \subset \mathcal{N}$. Since $(\pi^*, \lambda^*, \omega^*, \sigma^*)$ is an optimal solution to the dual of the linear program for set $\mathcal{N}$, it will be feasible for the dual of the corresponding program for set $S$. (This is easy to see since constraints in (36) remain the same, the right hand side for some edges in (37) reduces from $R^{(o,d,i)} - c_e^c$ to $-c_e^c$ and constraints in (38) either remain the same or drop out as the cycles become infeasible for carriers in $S$). Since the dual is a minimization problem,

$$\begin{aligned} r(opt(S)) &\leq \sum_{k \in S} \left( \sum_{(o,d,i) \in \Theta_k} \omega^{(o,d,i)^*} D^{(o,d,i)} + \sum_{a \in \mathcal{A}} \sigma_a^* N_k^a \right) \\ \Rightarrow r(opt(S)) &\leq \sum_{k \in S} x_k. \end{aligned}$$

□

**Theorem 2.** *The core of the sea cargo network design game is non-empty if the LP relaxation has no integrality gap.*

Since the payoff vector in (40) is in the core, this theorem easily follows. The following example shows that the converse of the theorem does not hold in general.

Consider an instance with three ports with distance (in terms of the number of weeks that a ship takes to reach from one port to another) as depicted in Figure 12. Consider two carriers $A$ and $B$ with two ships each. Carrier $A$ has 1000 TEUs of demand from port $P_1$ to $P_2$ and 1000 TEUs of demand from port $P_1$ to port $P_3$. Similarly, carrier $B$ has 1000 TEUs of demand from port $P_3$ to $P_2$ and 1000 TEUs of demand from port $P_3$ to port $P_1$. Assume all ships are identical with 1000 TEU capacity, the revenue generated by satisfying a unit of demand depends on the distance between ports (in direct proportion i.e. revenue generated by satisfying a unit demand at port $P_1$ from port $P_2$ is 1 unit and the revenue generated by satisfying a unit demand at port $P_1$ from port $P_3$ is $\sqrt{2}$ units) and there are no costs involved. Clearly,

$r(opt(\{A\})) = \$1000$: 2 ships (1 each week) on cycle $P_1 - P_2 - P_1$.

$r(opt(\{B\})) = \$1000$: 2 ships (1 each week) on cycle $P_3 - P_2 - P_3$.

$r(opt(\{A, B\})) = \$2\sqrt{2} * 1000$: 3 ships (1 each week) on cycle $P_1 - P_3 - P_1$.

LP relaxation $r(opt(\{A, B\})) = \$2\sqrt{2} * 1000 + 0.5 * 1000$: 3 ships (1 each week) on cycle $P_1 - P_3 - P_1$ and 1 ship (0.5 each week) on either cycle $P_1 - P_2 - P_1$ or cycle $P_3 - P_2 - P_3$.

In this example, even though there is an integrality gap (LP relaxation $r(opt(\{A, B\}))$ - $r(opt(\{A, B\})) = 500$), a payoff in the core exists $((\$\sqrt{2} * 1000, \$\sqrt{2} * 1000))$.

## *3.6 Conclusions*

In this chapter we studied alliance formation among carriers. First, we presented the motivation behind alliance formation. Carriers collaborate with each other to share costs, extend customer base, provide better service frequency, achieve economies of scale and to better utilize their assets. Then, we considered the distribution of the overall revenue generated by an alliance among its members. For a special case of the problem, when the LP relaxation of the problem has no integrality gap, we provided

an allocation of the benefits that is in the core.

# CHAPTER IV

# MECHANISM DESIGN FOR A MULTICOMMODITY FLOW GAME IN SERVICE NETWORK ALLIANCES

## *4.1 Introduction*

In the transportation industry carriers form alliances to design collaborative service networks. They bring their assets, such as ships in sea cargo transportation, in to a pool and operate them together to provide capacity on the edges of the network. Further, carriers use this capacity to route their demand in the network. Collaborative networks help carriers to improve asset utilization, share capital costs, gain economies of scale and explore new markets. This situation where multiple participants/players utilize the underlying collaborative network to simultaneously route their revenue maximizing demand appears in other application areas as well. In the case of the internet, the backbone infrastructure and network is provided by a set of companies, and businesses and end users utilize the capacity on the network to route their data by paying fees to the network owner(s). Similarly, in procurement networks where multiple commodities are sourced through common suppliers, allocating capacity appropriately among the several commodities assumes great importance in determining the profitability of the entire system [13].

In a centralized setting, for instance when there is a single decision-maker who makes all the decisions, a *maximum multi-commodity flow* problem can be solved to identify the set of demands to satisfy and to route the demand on the network. The objective in this problem is to maximize the revenue by simultaneously shipping different commodities in such a way that the total amount of flow through each edge

is no more than the edge's capacity. However, in real life applications where multiple players interact, although individual players form alliances to take advantage of the synergies that exist among their operations, usually only part of the decisions regarding the alliance's operations are determined jointly. For example, given the capacities in the underlying collaborative network, individual carriers selfishly make their own routing decisions to satisfy their demand and maximize their own revenue. Also, they collect the revenue generated by satisfying their own demand and exchange capacity on the assets at some pre-determined prices. Specifically, a player does not allow other players to freely utilize his capacity. In this chapter, we provide a mechanism, an algorithmic solution adorned with side payments, to regulate the interaction among the players by determining capacity exchange costs. These exchange prices guide the individual players' profit maximizing behavior towards the solution of the centralized multicommodity flow problem. Furthermore, they lead to allocation of the total revenue among the players with desirable properties.

Flow games were first considered by [37] and [38] for networks with a single commodity where each arc is owned by an unique player. [21] extended the results of [38] by studying flow of multiple commodities on networks with a common source and a sink for all commodities where each arc again assumed to be owned by an unique player. More recently, [42] studies the coalitional multicommodity game in which players are the nodes of the given network who own capacity in terms of the maximum flow allowed through that node. In general, the focus of these papers is to first obtain the optimal flow in the network by solving the centralized problem using classical linear programming methods and then allocate the revenue among the players in a "fair" way. To define fairness, most of the work in literature utilizes the notion of the *core* allocation, the set of stable and budget balance allocations. [38] and [42] utilize the dual solution to show that the core is non-empty for the particular flow games they consider. In general, [48] shows that for any linear programming

game dual solution provides an allocation in the core.

The strategy of first computing a centralized solution to a game and then allocating the revenues obtained by this solution assumes a centralized planner in the sense that, the individual players have no control on the system operations and at the end they receive an overall payoff from the central planner. The centralized systems are very efficient as the decision-maker chooses the optimal solution for the collaboration and allocates the overall benefits in a fair manner so that the collaboration is stable. However, in most settings designing fully centralized systems are not realistic. For example, in carrier alliances the individual carriers will operate their own assets and route their own demand incurring the relevant costs and revenues. Given that centralization is not always possible, in decentralized system the incentives within the system must be designed in such a way that the individual players are motivated to chose solutions that are collectively optimal for the collaboration and the revenue obtained is close to the maximum revenue that can be obtained with a fully centralized system. Decentralized shortest path problems have been studied from a mechanism design point of view by for example [46] and [33]. They design second price auctions or VCG type payment mechanisms to evaluate the cost of an edge in the network. A well known problem with VCG auctions however is that the resulting prices may not be in the core.

In this chapter, we consider a more general network as compared to the literature in the sense that we allow multiple players to own capacity on a single edge. Our model is further differentiated from the ones studied by [38] and [21] since we consider multiple commodities with their respective sources and sinks. More over, the contribution of this chapter is the design of a mechanism to distribute the benefits of collaboration among the players in a decentralized setting of the multicommodity flow game. As the players are selfish, an individual player follows a strategy which maximizes his own revenue. We model this strategy explicitly as a linear program

($LP$). The mechanism drives each individual player's $LP$ towards the collaborative optimal solution using inverse optimization techniques. Specifically, the mechanism computes *capacity exchange costs* on the edges of the network so that given these costs the routing and capacity exchange decisions each player makes selfishly results in the collaborative optimal flow. The mechanism allows a player to collect the revenue from satisfying his own demand and charge (pay) other players for utilizing capacity owned by him (them). We show that for the decentralized multicommodity flow game, capacity exchange costs can always be found efficiently. Further, for a special case of the problem, i.e. when each edge in the network has an unique owner, we prove that the net revenue received by the players operating under the mechanism provides an allocation in the core.

The rest of the chapter is organized as follows: in the next section we review some relevant game theoretic concepts. In Section 4.3 we present the formal problem definition. In Section 4.4 we provide a basic core allocation. The design of the mechanism and related results are discussed in Section 4.5. Our conclusions are presented in Section 4.6.

## *4.2 Game Theory Basics*

Two game theoretic concepts are relevant to this chapter - the notion of core and *mechanism design*. The definition of core is provided in Chapter 3. We now define mechanism design.

Given a set of players $N$ and a set $\mathcal{O}$ of possible outcomes for these players, we let $v^i(o)$ be the valuation of outcome $o$ for player $i$. The goal of *mechanism design* is to design an algorithm that chooses an outcome, $\tilde{o} \in \mathcal{O}$ and an n-tuple of side payments $\{s^1, \ldots, s^n\}$ such that the total payment, $x^i$, to player $i$ is $x^i = v^i(\tilde{o}) + s^i$. The total payment is what each individual player aims to optimize. Intuitively, a mechanism design concerns both an algorithmic ingredient (to obtain the desirable solution) and

a payment ingredient that motivates the players. This field has received widespread attention recently and has been used successfully to develop algorithms and protocols for inter-connected collection of computers such as on the internet. For a detailed discussion of mechanism design we refer the reader to [46].

## *4.3 Problem Definition*

We define the maximum multicommodity flow ($MCF$) game as follows: let $G = (V, E)$ be a directed graph on a set of nodes $V$ and a set of edges $E$. Let $N$ be the set of players. Each player $i \in N$ has a demand set $D^i$ where $d_{(o,d,i)}$ denotes the maximum amount of existing demand from node $o$ to node $d$ for player $i$ and $r_{(o,d,i)}$ denotes the revenue generated by delivering a unit of demand from $o$ to $d$ by player $i$. Let $\cup_{i \in N} D^i = D$. Each edge $e \in E$ has a capacity $c_e$, which denotes the maximum amount of flow, summed over all commodities, allowed on that edge. $\gamma_e^i$ denotes the fraction of capacity that player $i$ owns on edge $e$ and $\sum_{i \in N} \gamma_e^i = 1$. In the multicommodity setting, each demand triplet $(o, d, i)$ is referred to as a different commodity. An outcome of the game is a feasible multicommodity flow, subject to demand and capacity constraints, that maximizes the overall revenue generated, denoted by $opt(N)$, and an allocation, $x \in \mathbb{R}^n$, of the total revenue among the players. For a set $S \subset N$, we denote by $opt(S)$ the maximum revenue generated by the network induced by players in set $S$.

Next, we present a linear program to compute the flow that maximizes the revenue generated by the grand coalition. For ease of exposition, we introduce fictitious edges $(d, o, i)$, from node $d$ to node $o$, for every commodity $(o, d, i)$ such that the only flow allowed on edge $(d, o, i)$ is for commodity $(o, d, i)$. For each $(o, d, i)$, $f_e^{(o,d,i)}$ denotes the amount of flow of commodity $(o, d, i)$ on edge $e$. The linear program can be written as:

$$(P^N): \quad opt(N) = \max \sum_{(o,d,i) \in D} f_{(d,o,i)}^{(o,d,i)} r_{(o,d,i)} \tag{41}$$

such that

$$\sum_{\{e:e\in IEdges(v)\}} f_e^{(o,d,i)} - \sum_{\{e:e\in OEdges(v)\}} f_e^{(o,d,i)} \leq 0 \qquad \forall v \in V \;\; \forall (o,d,i) \in D \quad (42)$$

$$\sum_{(o,d,i)\in D} f_e^{(o,d,i)} \leq c_e \qquad \forall e \in E \quad (43)$$

$$f_{(d,o,i)}^{(o,d,i)} \leq d_{(o,d,i)} \qquad \forall (o,d,i) \in D \quad (44)$$

$$f \geq 0. \quad (45)$$

In $(P^N)$, $IEdges(v)$ denote the set of incoming edges into node $v$ and $OEdges(v)$ denote the set of outgoing edges from node $v$. The objective function (41) maximizes the revenue obtained by satisfying demand. Constraint (42) is a flow balance constraint at every node of the network. Note that if these inequalities hold for each node $v \in V$ then in fact they must hold with equality at every node. Constraint (43) is a capacity constraint on each edge and constraint (44) is a demand constraint for every commodity.

Let $(D^N)$ denote the dual of the linear program $(P^N)$. Let $\pi = \{\pi_v^{(o,d,i)} : \pi \geq 0, \; \forall v \in V, \; \forall (o,d,i) \in D\}$, $\alpha = \{\alpha_e : \alpha_e \geq 0, \; \forall e \in E\}$ and $\beta = \{\beta_{(o,d,i)} : \beta_{(o,d,i)} \geq 0, \; \forall (o,d,i) \in D\}$ be the dual variables associated with constraints (42), (43) and (44) respectively.

## *4.4 An Allocation in the Core*

For cooperative games when the players are concerned with an overall payoff only, utilizing the dual optimal solution to the grand coalition's optimization problem is a common strategy to obtain an allocation for the players. Such a payoff vector is called a *dual payoff* vector and is obtained by assigning to each player an amount which corresponds to the value of his resources, according to the dual solution. Also, it is a well known result that for linear programming games, such as the one we consider here, the dual payoff vectors provide an allocation in the core [48]. [38] showed that for simple networks (single commodity flow networks where each edge

has a single unit of capacity), every core allocation corresponds to an optimal dual solution for the corresponding optimization problem. [56] proved a similar result for a more general class of linear programming games.

**Theorem 3.** *The core of the multicommodity flow game defined above is non-empty and a payoff allocation in the core can be constructed in polynomial time by solving the dual program* $(D^N)$*. However, the core of this game is not fully characterized by these dual solutions.*

*Proof.* This theorem can be easily proven by considering an optimal dual solution $(\pi, \alpha, \beta)$ and a payoff vector $x = \{x^1, \ldots, x^n\}$ such that the payoff to player $i$ is:

$$x^i = \sum_{e \in E} \gamma_e^i c_e \alpha_e + \sum_{(o,d,i) \in D^i} d_{(o,d,i)} \beta_{(o,d,i)}. \tag{46}$$

Next, we provide an example to show that the core of the $MCF$ game cannot be fully characterized by the dual payoff vector (46). Consider a network with two nodes 1 and 2 and edge (1,2). Consider two players $A$ and $B$ such that player $A$ owns 2 units of capacity on edge (1,2) and player $B$ has a unit demand from 1 to 2 generating a unit revenue. Clearly, the optimal solution value is one unit and $(x^A = 0.5, x^B = 0.5)$ is an allocation in the core. However this payoff vector does not correspond to any dual payoff vector (46) since every dual payoff vector will assign zero units to player $A$. □

## *4.5 Mechanism Design*

In a $MCF$ game, players collaborate by sharing capacity on the edges of the network. Let the optimal collaborative flow in the network be $f^*$, an optimal solution to $(P^N)$. In many applications, given an optimal flow in the network, players are concerned with not only their overall payoffs but also the flow decisions. For example when carriers collaborate in transportation networks, each carrier collects the revenue from his customers by satisfying their demand and evaluates the valuation of solution $f^*$

for himself as $v^i(f^*) = \sum_{(o,d,i)\in D^i} r_{(o,d,i)} f^{*(o,d,i)}_{(d,o,i)}$. Note that it is in the best interest of the collaboration that players make their demand routing and capacity exchange decisions as implied by $f^*$. However, the computation of optimal flow $f^*$ ignores the ownership of capacity on the edges by the players. That is, it assumes the players readily exchange capacity on the edges of the network.

We now present a mechanism that provides incentives to the players to route their flows and exchange capacity among themselves as prescribed by $f^*$. The incentive is provided in the form of capacity exchange costs, denoted by the vector *cost*. Capacity exchange costs provide side payments, on top of the valuations $v^i(f^*)$, to the players to guide their selfish behaviors towards the optimal solution $f^*$. That is, the net profit generated by a player $i$ is calculated by adding $v^i(f^*)$ to the money player $i$ receives from the other players using capacities on his edges minus the money $i$ pays to others for using their capacities. We present a linear program, representing the perspective of an individual player, that makes routing decisions for each player in order to maximize the net profit for the player. Further, we utilize inverse optimization techniques and calculate capacity exchange costs so that $f^*$ becomes the optimal solution for all the individual player's $LP's$.

### 4.5.1 Single Player Problem

The key difficulties in modelling the individual player's perspective in an alliance are in determining: how should the player account for the flow of other players in the network and what portion of their capacities can he use for routing his own flow. We model the individual perspective of a player as a linear program. We assume that if a player $i$ owns $\gamma^i_e$ fraction of capacity on edge $e$ then he is allowed to collect $\gamma^i_e$ fraction of the cost payed by other players for using capacity on edge $e$. We model an individual player's perspective by assuming that he can route all the flow in the network to maximize his profit. However in practice, an individual player

can only make decisions regarding his own flow. In this sense our approach is a conservative one, since the maximum revenue that player $i$ can obtain will always be less than the optimal value of this model. However, as described later, this model leads to allocations with desirable properties. In mathematical terms we represent the optimization problem solved by player $i$ in the alliance as:

$(PS^i)$ :

$$\max \sum_{(o,d,i)\in D^i} f^{(o,d,i)}_{(d,o,i)} r_{(o,d,i)} + \sum_{e\in E} cost_e \left[ \gamma^i_e \sum_{(o,d,i)\notin D^i} f^{(o,d,i)}_e - \left(1-\gamma^i_e\right) \sum_{(o,d,i)\in D^i} f^{(o,d,i)}_e \right] \quad (47)$$

such that

$$\sum_{e:e\in IEdges(v)} f^{(o,d,i)}_e - \sum_{e:e\in OEdges(v)} f^{(o,d,i)}_e \leq 0 \;\; \forall v \in V \qquad \forall (o,d,i) \in D \qquad (48)$$

$$\sum_{(o,d,i)\in D} f^{(o,d,i)}_e \leq c_e \qquad \forall e \in E \qquad (49)$$

$$f^{(o,d,i)}_{(d,o,i)} \leq d_{(o,d,i)} \qquad \forall (o,d,i) \in D \qquad (50)$$

$$f \geq 0. \qquad (51)$$

The objective function (47) consists of three terms. The first term maximizes the revenue generated by satisfying demand. The second term computes the cost paid to player $i$ by other players for using capacity on an edge owned by player $i$. Similarly, the third term represents the cost paid by player $i$ to other players for using capacity on their edges. Constraints (48)-(51) are network flow constraints.

### 4.5.2 Computation of Allocations Using Inverse Optimization

Our aim is to motivate player $i$ so that the collaborative solution $f^*$ is attractive to him. To this end, we wish to identify a vector *cost* such that $f^*$ is optimal for $PS^i$. The problem fits well in the inverse optimization framework where given a feasible solution (flow vector $f^*$) to the problem, we wish to identify the parameters of the problem (cost vector *cost*) which will make the given solution ($f^*$) optimal for the problem. Inverse problems have been studied in a variety of fields such as portfolio

optimization [20] and traffic equilibrium [22]. [3] provides a unified framework for studying many inverse optimization problems on networks such as the shortest path problem, the assignment problem and the minimum cut problem.

Let $(\pi^i)$, $(\alpha^i)$ and $(\beta^i)$ denote the dual variables associated with constraints (48), (49) and (50) respectively and $(DPS^i)$ denote the dual program corresponding to $(PS^i)$. Note that we use super-script $i$ to denote that the dual is corresponding to player $i$'s $LP$. One form of the linear programming optimality conditions states that the primal solution $f^*$ and a dual solution $(\pi^i, \lambda^i, \omega^i)$ are optimal for their respective problems if $f^*$ is feasible for $(PS^i)$ and $(\pi^i, \lambda^i, \omega^i)$ is feasible for $(DPS^i)$, and together they satisfy the complementary slackness conditions. We say that a cost vector, *cost*, is *inverse feasible* with respect to $f^*$ if $f^*$ is an optimal solution to $(PS^i)$ with cost vector *cost*. Let $\overline{E}$ denote the set of edges that are utilized at full capacity in flow $f^*$ (that is, $\overline{E} = \{e \in E : \sum_{(o,d,i)} f_e^{*(o,d,i)} = c_e\}$), $\overline{D}$ denote the set of demands that are fully satisfied (that is, $\overline{D} = \{(o,d,i) \in D : f_{(d,o,i)}^{*(o,d,i)} = d_{(o,d,i)}\}$), and $\overline{F}$ denote the set of non zero flow vectors (that is, $\overline{F} = \{f_e^{*(o,d,i)} : f_e^{*(o,d,i)} > 0\}$). Using the above notation following gives us a characterization of the inverse feasible cost vector for player $i$:

$$(I^i) : (\pi^i)_v^{(o,d,i)} - (\pi^i)_u^{(o,d,i)} + (\alpha^i)_e \geq -(1-\gamma_e^i)cost_e \;\; (o,d,i) \in D^i \; e \in E : f_e^{*(o,d,i)} \notin \overline{F} \quad (52)$$

$$(\pi^i)_v^{(o,d,i)} - (\pi^i)_u^{(o,d,i)} + (\alpha^i)_e = -(1-\gamma_e^i)cost_e \;\; (o,d,i) \in D^i \; e \in E : f_e^{*(o,d,i)} \in \overline{F} \quad (53)$$

$$(\pi^i)_v^{(o,d,i)} - (\pi^i)_u^{(o,d,i)} + (\alpha^i)_e \geq \gamma_e^i cost_e \;\; (o,d,i) \notin D^i \; e \in E : f_e^{*(o,d,i)} \notin \overline{F} \quad (54)$$

$$(\pi^i)_v^{(o,d,i)} - (\pi^i)_u^{(o,d,i)} + (\alpha^i)_e = \gamma_e^i cost_e \;\; (o,d,i) \notin D^i \; e \in E : f_e^{*(o,d,i)} \in \overline{F} \quad (55)$$

$$(\pi^i)_o^{(o,d,i)} - (\pi^i)_d^{(o,d,i)} + (\beta^i)_{(o,d,i)} \geq r_{(o,d,i)} \;\; (o,d,i) \in D^i : f_{(d,o,i)}^{*(o,d,i)} \notin \overline{F} \quad (56)$$

$$(\pi^i)_o^{(o,d,i)} - (\pi^i)_d^{(o,d,i)} + (\beta^i)_{(o,d,i)} = r_{(o,d,i)} \;\; (o,d,i) \in D^i : f_{(d,o,i)}^{*(o,d,i)} \in \overline{F} \quad (57)$$

$$(\pi^i)_d^{(o,d,i)} - (\pi^i)_o^{(o,d,i)} + (\beta^i)_{(o,d,i)} \geq 0 \;\; (o,d,i) \notin D^i : f_{(d,o,i)}^{*(o,d,i)} \notin \overline{F} \quad (58)$$

$$(\pi^i)_o^{(o,d,i)} - (\pi^i)_d^{(o,d,i)} + (\beta^i)_{(o,d,i)} = 0 \;\; (o,d,i) \notin D^i : f_{(d,o,i)}^{*(o,d,i)} \in \overline{F} \quad (59)$$

$$(\alpha^i)_e = 0 \; \forall e \notin \overline{E} \quad (\beta^i)_{(o,d,i)} = 0 \; \forall (o,d,i) \notin \overline{D} \quad (60)$$

$$(\pi^i)_v^{(o,d,i)} \geq 0 \; \forall v \in V, \; \forall (o,d,i) \in D \; ; \quad (\alpha^i)_e \geq 0 \; \forall e \in E \; ; \quad (\beta^i)_{(o,d,i)} \geq 0 \; \forall (o,d,i). \quad (61)$$

Thus the inverse problem for player $i$ is to find a dual solution and a cost vector that satisfies constraints (52) -(61). Constraints (52) - (55) are the dual feasibility and complementary slackness conditions for the flow variables corresponding to each demand triplet and each edge of the network. The first two constraints correspond to the demand owned by player $i$ and the next two constraints correspond to the demand owned by the other players. Similarly, constraints (56) - (61) are the dual feasibility and complementary slackness conditions for the flow variables corresponding to demand triplets and the fictitious edges of the network. Constraints (56) - (59) correspond to the demand owned by player $i$ and the last two constraints are for the demand owned by other players. Recall that our aim is to determine a cost vector such that the flow $f^*$, as given by the optimal solution to $(P^N)$, is optimal for all single player problems $(PS^i)$. Extending the above arguments for all the players gives us the following characterization of the inverse feasibility problem to determine the vector *cost*

$$(INVP): \quad \cup_{i \in N} (I^i). \tag{62}$$

Thus the inverse problem $INVP$ is the union of all the individual players' inverse problems connected together with the common *cost* vector.

To prohibit players from colluding to form sub coalitions, we require a cost vector such that for any subset $S \subset N$ of players $f^*$ is an optimal solution for the corresponding problem $PS^S$. However, there are an exponential number of such subsets and including the inverse problem corresponding to each of them in $INVP$ will cause $INVP$ to become exponential in size. The next theorem shows that it is sufficient to consider only individual players' problems in $INVP$ to determine a suitable cost vector.

**Theorem 4.** *The inverse problem $(INVP)$ identifies a cost vector such that $f^*$ is optimal for any $PS^S$ such that $S \subset \mathcal{N}$.*

*Proof.* Let $\left((\pi^{i*})_{i\in N}, (\alpha^{i*})_{i\in N}, (\beta^{i*})_{i\in N}, cost^*\right)$ denote a feasible solution to $INVP$. Consider a subset $S \subset N$. Our claim is that for the inverse problem, $I^S$, corresponding to the problem $PS^S$ modeling the selfish behavior of subset $S$, $\left(\sum_{i\in S}(\pi^{i*}), \sum_{i\in S}(\alpha^{i*}), \sum_{i\in S}(\beta^{i*}), cost^*\right)$ is a feasible solution. That is, $cost^*$ is inverse feasible for $I^S$. Let,

$$(\pi^{S*}) = \sum_{i\in S}(\pi^{i*})$$
$$(\alpha^{S*}) = \sum_{i\in S}(\alpha^{i*})$$
$$(\beta^{S*}) = \sum_{i\in S}(\beta^{i*}).$$

Consider a constraint similar to (52) for the inverse problem $I^S$. Let $(o,d,k) \in D^k,\ k \in S$ and $f_e^{*(o,d,k)} \notin \overline{F}$. Since $\left((\pi^{i*})_{i\in N}, (\alpha^{i*})_{i\in N}, (\beta^{i*})_{i\in N}, cost^*\right)$ is a feasible solution for $INVP$ we have:

$$(\pi^{k*})_v^{(o,d,k)} - (\pi^{k*})_u^{(o,d,k)} + (\alpha^{k*})_e \geq -(1-\gamma_e^k)cost_e^*$$
$$(\pi^{i*})_v^{(o,d,k)} - (\pi^{i*})_u^{(o,d,k)} + (\alpha^{i*})_e \geq \gamma_e^i cost_e^*\ \ i \neq k,\ i \in S.$$

After summing these inequalities and rearranging the terms we get:

$$\sum_{i\in S}(\pi^{i*})_v^{(o,d,i)} - \sum_{i\in S}(\pi^{k*})_u^{(o,d,i)} + \sum_{i\in S}(\alpha^{k*})_e \ \ \geq \ \ -(1-\sum_{i\in S}\gamma_e^i)cost_e^*. \tag{63}$$

If for demand $(o,d,k)$ and edge $e$, $f_e^{*(o,d,k)} \in \overline{F}$ then in $INVP$ constraints corresponding to $f_e^{*(o,d,k)}$ will be at equality and hence we will get equality in (63). Similarly, $\left((\pi^{S*}), (\alpha^{S*}), (\beta^{S*}), cost^*\right)$ satisfies other constraints in $I^S$. Since our choice of $S$ was arbitrary this holds for every subset $S \subset N$. Thus it is enough to consider only the inverse problems corresponding to individual players in $INVP$ to determine a cost vector that also prevents collusion among subsets of players. □

To summarize, the inverse problem is a feasibility problem to identify the cost vector $cost^*$ and the overall payoff assigned by our mechanism to player $i$ is:

$$x^i \ \ = \ \ \sum_{(o,d,i)\in D^i} r_{(o,d,i)} f_{(d,o,i)}^{*(o,d,i)} + s^i \tag{64}$$

where the vector of side payments is

$$s^i = \sum_{e \in E} cost_e^* \left[ \gamma_e^i \sum_{(o,d,i) \notin D} f_e^{*(o,d,i)} - \left(1 - \gamma_e^i\right) \sum_{(o,d,i) \in D^i} f_e^{*(o,d,i)} \right] \ \forall i \in N. \tag{65}$$

The vector of payoffs (93) is such that $\sum_{i \in N} x^i = opt(N)$. This is easy to see since once a feasible solution is found for $INVP$, the flow in the network is same as the optimal flow $f^*$. Also note that the vector of side payments $\{s^1, s^2, \cdots, s^n\}$ is such that $\sum_{i \in N} s^i = 0$. Next, we prove that the inverse problem $INVP$ is feasible and thus the mechanism can always be used to find a cost structure on the edges of the network.

**Theorem 5.** *The inverse problem $INVP$ is feasible.*

*Proof.* For every inverse problem $I^i$, let $(y^i)^{(o,d,k)}$ $\forall (o,d,k) \in D$ and $(f^i)_e^{(o,d,k)}$ $\forall e \in E$, $\forall (o,d,k) \in D$ denote dual variables for constraints corresponding to (56)-(59) and constraints (52)-(55) respectively. The dual for $INVP$ can then be written as:

$$(DINVP): \quad max \sum_{i \in N} \sum_{(o,d,i) \in D^i} r_{(o,d,i)} (y^i)^{(o,d,i)}$$

$$\text{such that} \sum_{e: e \in IEdges(v)} (f^i)_e^{(o,d,i)} - \sum_{e: e \in OEdges(v)} (f^i)_e^{(o,d,i)} + \chi^{(v,o)} (y^i)^{(o,d,i)} - \chi^{(v,d)} (y^i)^{(o,d,i)} \leq 0 \ \ \forall v \in V, \ \forall (o,d) \in D, \ \forall i \in N$$

$$\sum_{(o,d,i) \in D} (f^i)_e^{(o,d,i)} \leq 0 \ \ \forall e \in E, \ \forall i \in N$$

$$\sum_{i \in N} \sum_{(o,d,i) \in D^i} \left(1 - \gamma_e^i\right) (f^i)_e^{(o,d,i)} - \sum_{i \in N} \sum_{(o,d,i) \notin D^i} \gamma_e^i (f^i)_e^{(o,d,i)} \leq 0 \ \ \forall e \in E$$

$$(y^i)^{(o,d,i)} \leq 0 \ \ \forall (o,d,i) \in D, \ \forall i \in N$$

$$(y^i) \geq 0, \ (f^i) \geq 0 \ \ \forall i \in N.$$

$\chi^{(u,v)}$ is an indicator function such that $\chi^{(u,v)} = 1$ if $u = v$ and 0 otherwise. Clearly the zero vector is a feasible solution to $DINVP$ and the optimal objective value is also zero (note that the last two constraints imply that $(y^i)^{(o,d,k)} = 0$ for all feasible solutions of $DINVP$). Since $DINVP$ has a finite optimal value, from duality theory we get that its dual $INVP$ is feasible. □

### 4.5.3 Multicommodity Flow Game with a Unique Owner on Each Edge

In this section, we consider a special case of the multicommodity flow game in which for every edge of the network, the capacity on the edge belongs to an unique player. We denote by $E^i$ the set of edges owned by player $i$. For this case, Theorem 6 states that the payoff allocation (93) made by our mechanism is an allocation in the core.

**Theorem 6.** *If each edge of the network has an unique owner, the payoff vector* $x = \{x^1, \ldots x^n\}$ *where* $x^i$ *is given by (93) lies in the core of the multicommodity flow game.*

*Proof.* Recall that the payoff obtained by the mechanism is budget balance. Hence, we only need to show the stability of payoff for every subset $S \subset N$. Consider a subset $S$ of players. The maximum revenue that the players in $S$ can achieve on their own is denoted by $opt(S)$. It is obtained by considering a linear program $(P^S)$ (similar to program $(P^N)$ in Section 4.3) on the network induced by the players in set $S$. Let us denote the dual for linear program $(P^S)$ as

$$(D^S): \quad opt(S) = \min \sum_{i \in S} \sum_{e \in E^i} c_e \alpha_e + \sum_{i \in S} \sum_{(o,d,i) \in D^i} d_{(o,d,i)} \beta_{(o,d,i)} \tag{66}$$

$$\text{such that } \ \pi_v^{(o,d,i)} - \pi_u^{(o,d,i)} + \alpha_{(u,v)} \geq 0 \ \ \forall (u,v) \in \cup_{i \in S} E^i \ , \ (o,d,i) \in \cup_{i \in S} D^i \tag{67}$$

$$\pi_o^{(o,d,i)} - \pi_d^{(o,d,i)} + \beta_{(o,d,i)} \geq r_{(o,d,i)} \ \ \forall (o,d,i) \in \cup_{i \in S} D^i \tag{68}$$

$$\pi_v^{(o,d,i)} \geq 0 \ \forall v \in V, \ \forall (o,d,i) \in \cup_{i \in S} D^i \ ; \quad \alpha_e \geq 0 \ \forall e \in \cup_{i \in S} E^i \tag{69}$$

$$\beta_{(o,d,i)} \geq 0 \ \forall (o,d,i) \in \cup_{i \in S} D^i. \tag{70}$$

As proved in Theorem (4), $(\pi^{S*}, \alpha^{S*}, \beta^{S*}, cost^*)$ is a feasible solution for $I^S$. We will show that $(\pi^{S*}, \alpha^{S*}, \beta^{S*})$ is a feasible solution for $(D^S)$. Since $(\pi^{S*}, \alpha^{S*}, \beta^{S*})$ is feasible for $(I^S)$, for $(o,d,i) \in D^i, i \in S$ we have

$$(\pi^{S*})_o^{(o,d,i)} - (\pi^{S*})_d^{(o,d,i)} + (\beta^{S*})_{(o,d,i)} \geq r_{(o,d,i)}. \tag{71}$$

This is easy to see by considering constraints corresponding to constraints (56) and (57) in $I^S$. Similarly, in the network induced by the players in $S$, for an edge

$e \in E^k$, and $(o,d,i) \in D^i$ such that $k, i \in S$ we have (by considering constraints corresponding to constraints (52) and (53) in $I^S$)

$$(\pi^{S*})_v^{(o,d,i)} - (\pi^{S*})_u^{(o,d,i)} + (\alpha^{S*})_e \geq -\Big(1 - \sum_{j \in S} \gamma_e^j\Big) cost_e^*. \tag{72}$$

Since player $k$ is the unique owner of edge $e$, $\gamma_e^k = 1$ and thus right hand side in (72) is zero.

$$\Rightarrow (\pi^{S*})_u^{(o,d,i)} - (\pi^{S*})_v^{(o,d,i)} + (\alpha^{S*})_e \geq 0. \tag{73}$$

Equation (71) and (73) suggest that $(\pi^{S*}, \alpha^{S*}, \beta^{S*})$ satisfies (67)-(70).

$$\Rightarrow \sum_{i \in S} \sum_{e \in E^i} c_e (\alpha^{S*})_e + \sum_{i \in S} \sum_{(o,d,i) \in D^i} d_{(o,d,i)} (\beta^{S*})_{(o,d)} \geq opt(S). \tag{74}$$

Since $d_{(o,d)} \geq 0$, $c_e \geq 0$, $(\alpha^{S*}) \geq 0$ and $(\beta^{S*}) \geq 0$ we have

$$\begin{aligned} &\sum_{i \in N} \sum_{e \in E^i} c_e (\alpha^{S*})_e + \sum_{i \in N} \sum_{(o,d,i) \in D^i} d_{(o,d,i)} (\beta^{S*})_{(o,d,i)} \geq \\ &\quad \sum_{i \in S} \sum_{e \in E^i} c_e (\alpha^{S*})_e + \sum_{i \in S} \sum_{(o,d,i) \in D^i} d_{(o,d,i)} (\beta^{S*})_{(o,d,i)} \end{aligned} \tag{75}$$

but left hand side in (75) is same as $\sum_{i \in S} x^i$ from strong duality applied to $(PS^S)$ and $(DPS^S)$. From (74), (75) and above argument we get $\sum_{i \in S} x^i \geq opt(S)$. □

## *4.6 Concluding Remarks*

In this chapter we presented a mechanism that regulates the interactions among the players in a multicommodity flow game. This scheme can be used in transportation and communications systems when multiple businesses and organizations use a collaborative service network to deliver goods. The mechanism provides capacity exchange costs on the edges of the network to motivate a player who owns the capacity on an edge to sell it to a player who can utilize that capacity to satisfy his own demand. Also the mechanism allows players to keep the revenue obtained by satisfying their demand in a collaborative solution. In a special case, when each of the network edges

is owned by an unique player we show that the allocation made by our mechanism provides an allocation in the core.

We considered a particular behavioral model to capture the selfish perspective of an individual player. Other behavioral models have also been considered in [2] and [34]. Different behavioral models lead to different capacity exchange prices with different properties. For further details on comparison of different behavioral models we refer the reader to [2]. Whether the payoff given by (93) provides an allocation in the core or not in the general case, that is when the edges have multiple owners, remains an open question.

# CHAPTER V

# MECHANISM DESIGN FOR LINER SHIPPING ALLIANCES

## *5.1 Introduction*

The discussion in Chapter 3 on payoffs among carriers in an alliance guarantees a stable allocation only for a subclass of the sea cargo network design game (when the integer program (1) - (7) in Chapter 2 and its linear programming relaxation have the same objective function value). Also, it works on the assumption that the carriers are indifferent towards which demands are satisfied and which routes are scheduled. However, this is usually not the case in practice. A carrier spends his resources on maintaining and operating his ships and benefits from satisfying his own demand. In this chapter we develop a mechanism that utilizes ideas from game theory, economics and inverse optimization to allocate benefits and costs among carriers. In literature a wide variety of problems ranging from auctions, voting, internet routing, transportation and traffic routing, facility location [30], [54] e.t.c. have been studied in a distributed setting. The general problem studied involves interaction among self-interested players with their own goals and preferences, and the capability of manipulating the system. The goal is to develop algorithms and protocols, for a set of players, that perform well when the players behave selfishly. The notion from game theory most relevant to this chapter is that of "mechanism design" (introduced formally in Chapter 4).

In general, the field of mechanism design aims to study how privately known preferences of many people can be aggregated towards a "social choice". Intutively, a mechanism solves a given problem by providing side payments to players to assure

that the required output occurs, when players choose their strategies so as to maximize their own selfish profits. This field has received widespread attention recently and has been used successfully to develop algorithms and protocols for inter-connected collection of computers such as on the internet. In the general setting of mechanism design problems the utility or the valuation of an outcome for a player is private information of the player and the goal is to design mechanisms to promote truthful reporting of the utility among players. Most often the mechanism design theory ignores the computational aspect of the protocol and emphasizes on theoretical results that might be computationally challenging to achieve. For a detailed discussion of mechanism design we refer the reader to [46].

In this chapter, the focus is to design computationally efficient mechanisms to distribute "benefits" and "costs" among different sea carriers in a liner shipping alliance and motivate them to act in an optimal fashion. In literature a few references are available regarding qualitative study of liner shipping alliances, however a quantitative study is missing. In this chapter we provide a mathematical framework to study these alliances. Carriers form strategic alliances by pooling their fleets and operating them together to share capacity on the ships. In such alliances, the carriers decide on a set of service routes, assign their ships for operating the chosen routes and allocate each ship's capacity among the alliance members. As a result of these alliances and agreements, shipments arranged through one carrier may actually be moved by a ship operated by another carrier and alliance members can offer higher sailing frequencies than would be possible using only their own ships. Alliances are most common on deep sea routes such as the Asia-North America route that require a bigger commitment in terms of assets (ships) from carriers. In the mid 1990s an estimated 60% of the total global liner capacity was accounted by alliances.

From the perspective of a single carrier, each carrier would like to design a service network which maximizes his profit. However, since he is working in collaboration

with other carriers, a network that generates maximum overall revenue for all carriers is selected. Clearly, such a network can be generated by replacing the individual carriers by a large carrier, with demand equal to the consolidated demand of all carriers and fleet equal to the pooled fleet of all carriers, and obtaining a schedule (or outcome say $\tilde{o}$) by solving the optimization problem presented in Chapter 2 for the large carrier. We refer to this solution as *the collaborative optimal solution.* Each carrier has a valuation of the collaborative optimal solution for himself, depending on the cost incurred by him and the revenue generated by shipping his demand on the collaborative routes. Let us denote the valuation of schedule $\tilde{o}$ for carrier $k$ by $v_k(\tilde{o})$. $v_k(\tilde{o})$ alone is not guaranteed to provide enough motivation for carrier $k$ to act according to the schedule, $\tilde{o}$. The goal in this chapter is to obtain an n-tuple of side payments $\{s_1, \cdots, s_n\}$ such that for every carrier $k$ the total payoff to carrier $k$, $x_k = v_k(\tilde{o}) + s_k$, motivates him to act in accordance with the optimal schedule $\tilde{o}$. In the collaborative optimal solution often the "resource," e.g. capacity on a ship, belongs to some carrier who does not allow other carriers to freely obtain "benefits" from using it. The fair distribution of costs and benefits among the alliance members is an intriguing question and in transportation networks, in particular liner shipping, sharing benefits and costs among the carriers generally translates into exchanging asset capacity on service routes. One way to regulate capacity exchanges among the carriers is to assign suitable *capacity exchange costs* so that the carrier who has unused capacity on a ship is motivated to sell the capacity to a carrier who can utilize it to deliver cargo. We utilize capacity exchange costs to provide side payments to the carriers. These payments are made by the carriers who utilize a ship's capacity, depending on the amount of utilization, to the carrier who owns the ship. To compute these costs we model the individual behavior of a carrier in the alliance as a linear program. The mechanism drives each individual carrier's linear program towards the collaborative optimal solution $\tilde{o}$ using inverse programming techniques.

In the next section we present a brief literature review and in Section 5.3 we present the formal problem definition. In Section 5.4 we present our solution strategy and an inverse optimization based mechanism to compute side payments for the carriers. In Section 5.5 we compare the fairness of the payoff to a carrier obtained by our algorithm against the *core* allocation. As described formally in Chapter 3, allocations in the core are budget balance and stable. Finally, in Section 5.6 we present future directions for our research.

## *5.2 Literature Review*

There exists literature that joins cooperative game theory with classical routing problems. [57] develops an allocation mechanism for a transportation game. [32] and [31] study delivery games associated with the Chinese postman problem. [41] and [26] study the vehicle routing game with homogeneous and heterogeneous fleet respectively and allocate cost among the members of an alliance based on cooperative game theory concepts. Network related games such as the network design games [39], the assignment game [61] and the facility location games [30] have also been studied in the literature. Some studies on combinatorial games such as the bin packing games [25] and the knapsack game [25] are also available.

In particular, qualitative studies regarding alliance formation among carriers in liner shipping industry are available in the literature. [44], [63], [55], [64] discuss the importance of strategic alliances in liner shipping. In particular, [44] studies the factors that led to the advent of strategic alliances among liner carriers 30 years ago, and the changes in the industry in the 1990s (for example, the increase in demand due to globalization) that made the previous alliances inadequate and called for a new generation of strategic partnerships. It suggests that differentiation in the contribution of each member, depending on their core competencies, can lead to alliances that deliver more than the sum of individual contributions. Also, alliances with fewer

members or ones that are led by a dominant partner are more likely to succeed. [63] provides industry data to support the claim that alliances lead to intensification in service frequency and an increase in ship size. It points out that as a result of alliances, carriers are becoming more similar (with similar service routes, serving the same markets and employing comparable ships) and although individual carriers who form alliances serve more ports than before, the total number of ports served by the overall industry remain remarkably constant. [55] studies the progression of collaborations from consortia, which are route-based forms of cooperation, to alliances, which cooperate on a global level, among Asian carriers. They argue that the reasons behind this trend are the flexibilities and synergies provided by alliances in global perspective. [64] makes use of cooperative game theory and provides a quantitative study to analyze liner shipping alliances by considering two small examples (3 ports and 2-3 carriers). They explicitly write all the core inequalities to analyze the alliance and allocate the revenue among the members in ratio of their shipping capacity. However, as we will see later, in many situations such a proportional allocation of benefits is not guaranteed to provide a payoff to the alliance members to sustain the existence of the collaboration. Also, [64] does not provide any framework for dealing with larger instances for which it becomes harder to explicitly write all the core inequalities.

## *5.3 Problem Definition*

In this section we formally present our problem. Our attempt is to use as much as possible the standard notions from both mechanism design and inverse optimization. We will use the notation introduced in Chapter 2 and Chapter 3 to refer to the set, $\mathcal{N} = \{1, 2 \cdots n\}$, of carriers and their fleet size and demand triplets. To simplify our study we assume that all ships are identical. Thus for a carrier $k \in \mathcal{N}$, $\Theta_k$ represents his demand set and $N_k$ represents the number of ships in his fleet.

The *liner shipping mechanism design problem* is defined as follows. We are given

a set of carriers $\mathcal{N}$ who wish to form an alliance by consolidating their demand $\Theta = \cup_k \Theta_k$, and pooling their ships, $N = \sum_k N_k$. Following are the set of problems that the carriers face:

1. Together they need to design their service network. For this, they need to design a set of service routes (say $\overline{C} = \{C_1, \cdots, C_r\}$) to operate, utilizing their ships. Also, they need to decide a set of cargo (say $\overline{\Theta} \subset \Theta$) to deliver and the paths to use to deliver the selected cargo.

2. The members of the alliance need to decide how to realize the service routes in $\overline{C}$. For example, they need to decide the number of ships that each carrier should assign to the service routes in $\overline{C}$.

3. Each carrier $k$ needs to compute the valuation, $v_k$, of the solution, given by $(\overline{C}, \overline{\Theta})$, depending on the cost incurred by him and the revenue generated by delivering his demands.

4. For a given $(\overline{C}, \overline{\Theta})$, the valuation $v_k$ alone is not enough to guarantee that carrier $k$ will route his cargo and share capacity as determined by the collaborative solution. Thus the alliance needs to put in place a mechanism that provides the right incentives to the carriers in order for them to act as prescribed by the collaborative solution. We provide incentives in the form of side payments $\{s_1, \cdots, s_n\}$, such that the total payment, $x_k = v_k + s_k$, to carrier $k$ motivates him to participate in the alliance. Furthermore, the side payments are the net sum of the capacity exchange costs a carrier pays to and receives from other carriers.

## *5.4 Solution Strategy*

In this section, we propose a set of algorithms for resolving the above problems faced by the alliance members. Before providing the details, we first present an outline of

our solution strategy. The goal of an individual carrier is to design a service network which maximizes his profit. However, since he is working in collaboration with other carriers, a network that generates maximum overall revenue for all the carriers is selected. Clearly, such a network can be obtained by replacing the individual carriers' fleet by $N$, the combined fleet of all the carriers, and the individual demand sets by $\overline{\Theta}$, the combined demand of all the carriers, and then solving a network design problem as presented in Chapter 2 on this input. We use the solution strategy described in Chapter 2 to solve this optimization problem and obtain a collaborative optimal solution $opt(\mathcal{N}) = (\overline{C}, \overline{\Theta})$.

It is non-trivial for the alliance to realize the solution given by the optimization algorithm. In a carrier alliance such as the one addressed here, it is not reasonable to assume that there exists a fully centralized body that can operate the combined fleet, make the overall cargo accept-reject and routing decisions and hence gather the total profit and then allocate it among the members of the alliance in a fair manner. A more realistic model of operation is to assume that once the collaborative optimal service routes and the ships to operate on them are decided centrally, then the carriers individually operate their ships incurring the operational costs and make their own cargo accept-reject and routing decisions determining the revenue they earn. In such a partial decentralized system the challenges are (i) to design a mechanism that regulates the interactions among the carriers, that is exchange of capacity on each others' ships; and (ii) to provide incentives so that the carriers' individual accept-reject and routing decisions are as prescribed by the collaborative optimal solution.

One straight forward way to handle the exchange of capacity among the carriers is to use a proportional space allocation algorithm assigning each carrier a capacity on a network edge in proportion to the capacity provided by the carrier to that edge. However, such a simple allocation algorithm does not guarantee to provide outcomes that can achieve the generation of the maximum possible profit, $opt(\mathcal{N})$. To see this

consider a simple network with two ports $P_1$ and $P_2$ such that it takes a ship one week to reach from one port to the other. Now let two carriers $A$ and $B$ with one ship each of 1000 TEU capacity form an alliance to operate a service route with weekly frequency between the two ports. Let $A$ have 700 TEUs of demand and $B$ have 300 TEUs of weekly demand from $P_1$ to $P_2$. Assume that a unit amount of revenue can be generated by satisfying any demand and that ship operation costs are negligible. Then a proportional space allocation algorithm would assign a capacity of 500 TEUs to each carrier and would be able to generate only 800 units of weekly revenue whereas the optimal solution to the problem is 1000 units of weekly revenue.

The valuation of solution $opt(\mathcal{N})$ is calculated for each carrier by calculating the revenue generated by him and the costs incurred by him. The valuation obtained from solution $opt(\mathcal{N})$ however is not guaranteed to provide enough motivation for a carrier to act according to the schedule $opt(\mathcal{N})$. To provide this guarantee incentives in the form of side payments must be provided to the carriers.

To handle both of the challenges described above, we suggest that the centralized authority also determines capacity exchange costs on each edge of the service network and then lets the carriers make cargo accept-reject and routing decisions, buying and selling capacity along the way at the given prices. That is, we determine side payment as the net sum of the capacity exchange costs a carrier receives from others for utilizing capacity on his ships and pays to others for utilizing their capacity.

Computation of capacity exchange costs is however non-trivial. In Section 5.4.1 we show that the assignment of different carriers' ships to various selected cycles reduces to a generalized assignment problem in our setting. For a carrier, given an assignment of his ships to the cycles in $\overline{C}$, the computation of the fraction of capacity that the carrier owns on an edge is explained in Section 5.4.2. We denote by $cost_e$ the cost of using one TEU capacity on edge $e$. In other words, if a carrier provides capacity on an edge $e$ then for a unit utilization of capacity on edge $e$ he will charge

other carriers $cost_e$ times the fraction of the capacity he owns on edge $e$. Note that the optimal solution $opt(\mathcal{N})$ provides a flow vector $\overline{f}$ on the edges of the network. We provide a linear program that models the behavior of each carrier $k$ for making cargo accept-reject and routing decisions, equivalently determining his optimal flow vector $\overline{f^k}$. Finally, we use inverse optimization techniques to determine the cost vector, $cost_e$, such that the optimal flow vector $\overline{f^k}$ for each carrier $k$ is as prescribed by the collaborative optimal flow vector $\overline{f}$. That is, the vector of side payments determined by the capacity exchange costs makes the solution $opt(\mathcal{N})$ attractive to all the carriers.

We now provide the details of our solution strategy.

### 5.4.1 Valuation of the Schedule

For a carrier, the valuation of solution $opt(\mathcal{N})$ is determined by calculating the revenue generated by him and the costs incurred by him. The revenue generated by carrier $k$ is calculated by summing over the revenue generated by satisfying demand $(o, d, i)$ such that $(o, d, i) \in \overline{\Theta} \cap \Theta_k$. Similarly, each carrier pays for maintaining and operating his ships on the collaborative routes. To compute the costs incurred, a carrier first needs to know the assignment of his ships to the selected routes. Recall that as explained in Chapter 2, $L_C$ denotes the number of ships required to maintain a weekly frequency on cycle $C$ and that the cycle selection variables in the mixed integer linear program in (1) -(7) are binary i.e. the maximum frequency maintained on a cycle is a weekly frequency. Let, $y^k_{C_j}$ represents the number of ships carrier $k$ assigns to route $C_j$ and $u^k_{C_j}$ represents the utility he obtains by assigning one ship to route $C_j$. The problem of assigning ships, for all the carriers, to the set of selected service routes reduces to a generalized assignment problem which we refer to as the ship assignment problem

$(SAP)$.

$$(SAP): \quad \max \sum_{k,C_j} u^k_{C_j} y^k_{C_j} \tag{76}$$

$$\text{such that} \quad \sum_k y^k_{C_j} = L_{C_j} \qquad \forall C_j \in \overline{C} \tag{77}$$

$$\sum_{C_j} y^k_{C_j} \leq N^k \qquad \forall k \in N \tag{78}$$

$$y^k_{C_j} \quad int. \tag{79}$$

The generalized assignment problem is shown to be NP-hard. It is a well studied problem in literature and many heuristics have been proposed to solve it effectively [58]. The utility function, $u$, can be determined heuristically in many different ways. We compute the utility function in two different ways - first, we take the utility of assigning a ship to a service route for a carrier to be proportional to the flow of carrier on that service route and second, we take it to be proportional to the profit generated by the carrier from that service route. We solve the ship assignment problem exactly and heuristically. In Section 5.5.1 we report the effect of different ship assignment algorithms and different utility functions on the overall mechanism.

Once an assignment of ships to service routes is computed, the cost incurred by carrier $k$ in the alliance is computed as

$$Cost\ of\ operating\ routes = \sum_{C_j \in \overline{C}} \frac{Cost_{C_j}}{L_{C_j}} y^k_{C_j}$$

Note that cost of operating a ship on cycle $C_j$ (or the cost of cycle $C_j$ per week) is given by $Cost_{C_j}$. Thus if carrier $k$ assigns $y^k_{C_j}$ ships to cycle $C_j$ then the cost per week incurred by him on this cycle is $\frac{Cost_{C_j}}{L_{C_j}} y^k_{C_j}$. Finally, the valuation of solution $opt(N)$ for carrier $k$ is given by:

$$v_k(opt(\mathcal{N})) = \sum_{(o,d,i) \in \Theta_k} R^{(o,d,i)} \overline{f} - Cost\ of\ operating\ routes. \tag{80}$$

### 5.4.2 Computation of Side Payments

In an alliance, carriers work in collaboration with each other, however the primary objective of an individual carrier remains to be the maximization of his own profits. We model the selfish behavior of carriers by assuming that given the collaborative network the carriers solve their cargo routing problems individually. Given an assignment of ships, it is in the best interest of the collaboration that the carriers make their cargo routing decisions as in $\bar{f}$. Note that $\bar{f}$ requires carriers to share capacity on the ships. We facilitate this by allowing a carrier to charge other carriers for using capacity on the edges of the network. For this we need to determine when can a carrier charge other carriers for using capacity on an edge and at what price. Next, we answer these two questions.

We allow a carrier to charge other carriers for using capacity on a network edge $e$ whenever he owns the capacity on that edge i.e. whenever he has a ship assigned to $e$. The rest of the times he will need to pay other carriers for using capacity on edge $e$. As carriers pool their ships in an alliance to operate on service routes, usually multiple carriers have their ships assigned to any given edge. In other words, carriers usually own a fraction of the capacity on an edge. Given an assignment of ships of various carriers to the routes in $\overline{C}$, let $\gamma_e^k$ be the fraction of capacity that carrier $k$ owns on edge $e$. Recall that we assume all ships are identical with $T$ units of capacity. Thus, in the special case, when an edge $e$ is part of a single service route $C_j$ and the number of ships assigned by carrier $k$ to $C_j$ are $y_{C_j}^k$,

$$\gamma_e^k = \frac{y_{C_j}^k}{L_{C_j}}.$$

However, an edge can be part of many operated service routes. Next, we compute $\gamma_e^k$ for this case. Consider an edge $e$ that is part of multiple service routes, $\widetilde{C} = \{C_1, C_2, \cdots, C_r\} \subset \overline{C}$. The total capacity that carrier $k$ owns on edge $e$ due to his ships on service routes in $\widetilde{C}$ is $\sum_{C_j \in \widetilde{C}} \frac{T y_{C_j}^k}{L_{C_j}}$. As $e$ is part of $r$ cycles and thus total

capacity on $e$ is $Tr$, $\gamma_e^k$ is given by

$$\gamma_e^k = \frac{1}{r} * \sum_{C_j \in \widetilde{C}} \frac{y_{C_j}^k}{L_{C_j}}.$$

Clearly for an edge $e$, $\sum_{k \in \mathcal{N}} \gamma_e^k = \frac{1}{r} \sum_{k \in \mathcal{N}} \sum_{C_j \in \widetilde{C}} \frac{y_{C_j}^k}{L_{C_j}} = \frac{1}{r} \sum_{C_j \in \widetilde{C}} \sum_{k \in \mathcal{N}} \frac{y_{C_j}^k}{L_{C_j}} = \frac{1}{r} \sum_{C_j \in \widetilde{C}} 1 = 1$.

Once we have determined the fraction of capacity each carrier owns on an edge, the problem of determining suitable prices on the network edges reduces to the multi-commodity flow game discussed in Chapter 4. As denoted earlier, we refer to these prices as the capacity exchange costs and on an edge $e$ we denote it by $cost_e$. The capacity exchange costs provide side payments to the players, in addition to the valuation (80) obtained by them. Note that the capacity on the edges in the network is given by the collaborative optimal solution $opt(\mathcal{N})$, since carriers operate same service routes as in $opt(\mathcal{N})$. On an edge $e$, we denote this capacity by $\overline{Cap_e}$. Let $f^k = \{f_e^{(o,d,i),k} : f_e^{(o,d,i),k} \geq 0 \ \forall e \in E, \forall (o,d,i) \in \Theta\}$, where $f_e^{(o,d,i),k}$ represents the optimal flow of demand $(o,d,i)$ on edge $e$ when carrier $k$ makes his cargo accept-reject and routing decisions. Similar to Chapter 4 we propose the following mathematical formulation for modeling an individual carrier $k$'s behavior in the alliance:

$$(SCP^k):$$

$$\max \sum_{(o,d,i) \in \Theta_k} f_{(d,o)}^{(o,d,i),k} R^{(o,d,i)} + \sum_{e \in E_v} \Big( \sum_{(o,d,i) \notin \Theta_k} \gamma_e^k f_e^{(o,d,i),k} - \sum_{(o,d) \in \Theta_k} (1 - \gamma_e^k) f_e^{(o,d,i),k} \Big) cost_e \quad (81)$$

$$\text{such that} \sum_{e \in InEdges(v)} f_e^{(o,d,i),k} - \sum_{e \in OutEdges(v)} f_e^{(o,d,i),k} \leq 0 \quad \forall v \in V, \forall (o,d,i) \in \Theta \quad (82)$$

$$\sum_{(o,d,i) \in \Theta} f_e^{(o,d,i),k} \leq \overline{Cap_e} \quad \forall e \in E \quad (83)$$

$$f_{(d,o)}^{(o,d,i),k} \leq D^{(o,d,i)} \quad \forall (o,d,i) \in \Theta \quad (84)$$

$$f_e^{(o,d,i),k} \geq 0 \quad \forall e \in E, \forall (o,d,i) \in \Theta. \quad (85)$$

The objective function (81) consists of three terms. The first term denotes the revenue generated by satisfying demand corresponding to player $k$. The second term computes the cost paid to player $k$ by the other players for using capacity owned by

him. Similarly, the third term represents the cost paid by player $k$ to the other players in the alliance for using their capacity. Constraints (82)- (85) are the network flow constraints. Note that this is a conservative model since we allow an individual player to modify other player's flow also. In practice, an individual can only make decisions regarding his own flow. Thus the maximum revenue that player $k$ can obtain will always be less than the optimal value of $(SCP^k)$.

**Assignment of Prices to Network Edges** For the single carrier problem $(SCP^k)$ we wish to identify a cost vector, *cost*, such that the collaborative optimal flow, $\overline{f}$, is an optimal decision for the carrier also. This problem fits well in the inverse optimization framework where given a feasible solution (flow vector $\overline{f}$) to a linear program $(SCP^k)$, we wish to identify the parameters of the problem (cost vector *cost*) which will make the given solution (flow vector $\overline{f}$) optimal for the problem $(SCP^k)$. Next, we demonstrate the use of inverse optimization (as in Chapter 4) to compute the cost vector $cost_e$.

As in previous chapters, we denote by $\pi^k = \{\pi_v^{(o,d,i),k} : \pi_v^{(o,d,i),k} \geq 0 \quad \forall v \in V,\ \forall (o,d,i) \in \Theta\}$ , $\lambda^k = \{\lambda_e^k : \lambda_e^k \geq 0 \ \ \forall e \in E_v\}$ and $\omega^k = \{\omega^{(o,d,i),k} : \omega^{(o,d,i),k} \geq 0 \ \ \forall (o,d,i) \in \Theta\}$ the dual variables associated with constraints (82), (83) and (84) respectively. Note the use of super-script $k$ to denote that the dual is considered for the single carrier problem corresponding to carrier $k$. For carrier $k$, the dual of the $SCP^k$, denoted by $DSCP^k$, can be written as:

$(DSCP^k)$ :

$$\min \sum_{e \in E_v} \overline{Cap_e} \lambda_e^k + \sum_{(o,d,i) \in \Theta} \omega^{(o,d,i),k} D^{(o,d,i)} \tag{86}$$

such that

$$\pi^{(o,d,i),k}_{head(e)} - \pi^{(o,d,i),k}_{tail(e)} + \lambda^k_e \geq (\gamma^k_e - 1)cost_e \forall e \in E \setminus E_f, \forall (o,d,i) \in \Theta \setminus \Theta_k \quad (87)$$

$$\pi^{(o,d,i),k}_{head(e)} - \pi^{(o,d,i),k}_{tail(e)} + \lambda^k_e \geq \gamma^k_e cost_e \quad \forall e \in E - E_f \ , \ \forall (o,d,i) \in \Theta_k \quad (88)$$

$$\pi^{(o,d,i),k}_{head(e)} - \pi^{(o,d,i),k}_{tail(e)} + \omega^{(o,d,i),k} \geq 0 \quad \forall e \in E_f \ , \ \forall (o,d,i) \in \Theta \setminus \Theta_k \quad (89)$$

$$\pi^{(o,d,i),k}_{head(e)} - \pi^{(o,d,i),k}_{tail(e)} + \omega^{(o,d,i),k} \geq R^{(o,d,i)} \quad \forall e \in E_f \ , \ \forall (o,d,i) \in \Theta_k \quad (90)$$

$$\pi^k, \ \lambda^k, \ \omega^k \geq 0. \quad (91)$$

One form of the linear programming optimality conditions states that the primal solution $f^k$ and dual solution $(\pi^k, \lambda^k, \omega^k)$ are optimal for their respective problems if $f^k$ is feasible for the linear program in (82) - (85) and $(\pi^k, \lambda^k, \omega^k)$ is feasible for the linear program in (87) - (91), and together they satisfy the following complementary slackness conditions:

- For all edges $e$ that are not on full capacity, i.e. $\sum_{(o,d,i)} f^{(o,d,i),k}_e < \overline{Cap_e}$, $\lambda^k_e = 0$ for all carriers $k$.
- For all demand triplets $(o,d,i)$ that are not fully satisfied, i.e. $\sum_{j=1}^{7} f^{(o,d,i),k}_{(v_{(d,j)}, v_{(o,i)})} < D^{(o,d,i)}$, $\omega^{(o,d,i),k} = 0$ for all carriers $k$.
- For all edges $e$ with non zero flow, i.e. $f^k_e > 0$, the corresponding dual constraints in (87) -(91) are satisfied at equality.

We say that a cost vector *cost* is *inverse feasible* with respect to $f^k$ if $f^k$ is an optimal solution to $SCP^k$ with cost vector *cost*. Let us denote by $INVP^k$ the *inverse problem* that finds this cost vector. Note that $INVP^k$ is similar to the inverse problem $I^k$ in Chapter 4.

Our aim is to determine the cost vector such that the flow $\overline{f}$ as given by $opt(\mathcal{N})$ is optimal for all individual carrier problems i.e. $SCP^k \ \forall k \in \mathcal{N}$. From above, $\overline{f}$ is an optimal solution for every $SCP^k$ if for every $(DSCP^k)$ there exists a dual feasible

solution $(\overline{\pi}^k, \overline{\lambda}^k, \overline{\omega}^k)$ and a common cost vector *cost* that satisfies the primal-dual complementary slackness conditions. This gives us the following characterization of the inverse optimization problem we should solve to determine the vector *cost*:

$$(INVP): \quad \cup_{i \in \mathcal{N}} INVP^k \tag{92}$$

In other words, the inverse problem is a feasibility problem to identify the cost vector. If $\overline{cost}$ is a feasible solution to $(INVP)$, the overall payoff to player $k$ is given by:

$$x_k \;\; = \;\; v_k(opt(\mathcal{N})) + s_k \tag{93}$$

where the vector of side payments $\{s_1, s_2, \cdots, s_n\}$ is calculated as,

$$s_k = \sum_{e \in E} \Big( \sum_{(o,d,i) \notin \Theta_k} \gamma_e^k f_e^{(o,d,i)} - \sum_{(o,d,i) \in \Theta_k} (1 - \gamma_e^k) f_e^{(o,d,i)} \Big) \overline{cost}_e. \tag{94}$$

Note that the vector of payoffs to carriers $\{x_1, x_2, \cdots, x_n\}$ is such that $\sum_{k \in \mathcal{N}} x_k = r(opt(\mathcal{N}))$. This is easy to see since once a feasible solution is found for $(INVP)$, the flow in the network is the same as the flow of the optimal solution $opt(\mathcal{N})$. Also note that the vector of side payments $\{s_1, s_2, \cdots, s_n\}$ is such that $\sum_{k \in \mathcal{N}} s_k = 0$. Similar to Theorem 5 in Chapter 4, Theorem 7 guarantees that the inverse problem is feasible and a cost vector can always be found.

**Theorem 7.** *The inverse problem (INVP) is feasible.*

To prohibit the players from colluding to form sub-coalitions, we want to have a cost vector such that for any subset $S \subset \mathcal{N}$, $\overline{f}$ is an optimal solution for the corresponding problem $(SCP^S)$. However, there are exponential number of such subsets and including an inverse problem corresponding to each of them in $(INVP)$ will cause $(INVP)$ to become exponential in size. Theorem 4 from Chapter 4 guarantees that it is sufficient to consider only single carrier problems in $(INVP)$ to determine a cost vector that makes $\overline{f}$ optimal for any subset $S \subset \mathcal{N}$ of carriers.

It is reasonable to assume that an individual carrier would seek higher payoff in the alliance as compared to the revenue that he can generate on his own. To this end, we have enhanced our model by adding the following set of limited (single carrier and two carrier subset) rationality constraints in $(INVP)$:

$$opt(\{k\}) \geq x_k \text{ for each } k \in \mathcal{N}. \tag{95}$$

$$opt(\{k,i\}) \geq x_k + x_i \text{ for } k,i \in \mathcal{N}. \tag{96}$$

where, $opt(S)$ for $S \subset \mathcal{N}$ is the maximum revenue that the carriers in set $S$ can obtain, when working on their own.

## *5.5 Computational Experiments*

Next, we analyze liner shipping alliances from a quantitative as well as qualitative point of view. The focus of our computations is to study the performance of the mechanism designed in this chapter in the context of liner shipping.

We performed our computations on instances involving up to 10 ports with up to 27 demand triplets and 50 ships. The data is generated as explained earlier in Chapter 2 and the same notation is used to express various problem classes. All of our algorithms are implemented in C++ in an Unix environment. We also made extensive use of the callable libraries in CPLEX 9.0. All computational experiments were performed on a Sun280R system with UltraSparc-III processor. Results are reported on 50 randomly generated instances in each test class.

We measure the performance of the mechanism by checking if the payoffs obtained in (93) are in the core. As explained in Chapter 3, an allocation of benefits is in the core if it is budget balance and stable. Note that the core provides a very strong definition of stability where the grand alliance is not threatened by any of the sub-coalitions. However, as demonstrated by the example in Section 3.4 (Chapter 3), the core of the network design game in liner shipping can be empty.

### 5.5.1 Effect of Ship Assignment and Rationality Constraints

Recall that the problem of assigning ships of different carriers on the service routes is formulated as $(SAP)$ in Section 5.4.1. As noted earlier, this problem is NP-hard. However, in our case since the number of selected service routes are between 3 and 10 (depending on the problem size), an explicit enumeration scheme can also be used to determine the exact assignment of ships to the service routes. Next, we study how the mechanism is effected by different assignment of ships on the service routes. We obtain different assignment by considering different algorithms to solve $(SAP)$. In particular, we consider an exact assignment, a greedy assignment and a random assignment of the ships. We also consider two different utility functions ($u$ in the objective function of $(SAP)$) - 1. the utility of assigning a ship to a service route for a carrier is taken proportional to the sum of his flow on the edges of the service route (denoted by f) and 2. the utility of assigning a ship to a service route for a carrier is taken to be proportional to the sum of his profit generated from his flow on the edges of the service route (denoted by f.R).

We also study the effect of enhancing the inverse problem $(INVP)$ by adding rationality constraints. As mentioned at the end of Section 5.4.2, rationality constraint for a subset $S$ of carriers states that carriers in $S$ seek higher payoff in an alliance as compared to the payoff they can generate on their own. We divide the rationality constraints into different sets, depending on the number of carriers considered. For example, two carrier rationality constraints is the set of rationality constraints for all subsets with two carriers and is denoted by $\{2\}$. To study the effect of rationality constraints we introduce one set of rationality constraints at a time to the inverse problem $INVP$. Inverse program together with all the single carrier rationality constraints is denoted as $INVP + \{1\}$ and inverse program together with all the single carrier and two carrier rationality constraints is denoted as $INVP + \{1\} + \{2\}$. Recall that $INVP$ is a feasibility problem and the payoff allocation made by our algorithms

is always budget balance. Thus for a three carrier alliance if $INVP + \{1\} + \{2\}$ is feasible than it means that a cost structure that yields payoff allocation in the core can be identified.

Table 6 reports the effect of different assignment of ships on the service routes and the effect of rationality constraints on the solution quality for 3 carrier alliances and different test classes. In this table, we use $\{0\}$ to denote that $INVP$ is solved, $\{1\}$ to denote that $INVP + \{1\}$ is solved and $\{1\} + \{2\}$ to denote that $INVP + \{1\} + \{2\}$ is solved. The first column denotes different problem classes. It indicates the total number of ports, ships and demand pairs considered. Each demand and ship is assigned to one of the three carriers with equal probability. The next three columns report the number of instances (out of 50) for which an allocation in the core is found when the $SAP$ is solved exactly and the utility function is taken to be proportional to the flow times the revenue. The second column reports this number when the inverse problem $INVP$ is solved. The third, and fourth column report these numbers when the inverse problem ($INVP$) is solved together with all single carrier rationality constraints and ($INVP$) is solved with all single and two carrier rationality constraints, respectively. The next three triplet of columns report the corresponding numbers when $SAP$ is solved exactly, greedily and randomly, respectively. In these cases the utility function is taken to be proportional to the flow.

Different assignment algorithms and utility functions result in different number of ships being assigned by a carrier to each of the selected service routes. This in turn influences a carrier's valuation (80) of the optimal solution and the way the optimal solution is realized by the alliance. Note that the inverse problem computes the cost structure for a given assignment of ships to the service routes. Table 6 suggests that the number of cases for which the mechanism successfully finds a cost structure does not depend significantly on the assignment of ships to the service routes. If we consider all the rationality constraints, even for a random assignment of ships to the

**Table 6:** Effect of asset assignment and rationality constraints.

| Test Class | Exact | | | | | | Greedy | | | Random | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | f.R | | | f | | | f | | | f | | |
| | {0} | {1} | {1}+{2} | {0} | {1} | {1}+{2} | {0} | {1} | {1}+ {2} | {0} | {1} | {1}+{2} |
| P6S18D6 | 9 | 15 | 46 | 8 | 16 | 46 | 7 | 16 | 46 | 6 | 17 | 47 |
| P6S18D9 | 12 | 17 | 47 | 9 | 16 | 46 | 10 | 19 | 48 | 8 | 17 | 47 |
| P6S30D6 | 4 | 12 | 35 | 4 | 13 | 35 | 4 | 13 | 35 | 4 | 14 | 36 |
| P6S30D9 | 10 | 22 | 46 | 10 | 18 | 45 | 7 | 17 | 45 | 9 | 20 | 46 |
| P10S30D18 | 15 | 18 | 50 | 14 | 17 | 49 | 16 | 17 | 50 | 14 | 20 | 50 |
| P10S30D27 | 11 | 9 | 47 | 11 | 7 | 47 | 10 | 8 | 47 | 10 | 8 | 48 |
| P10S50D18 | 15 | 13 | 48 | 15 | 12 | 48 | 11 | 18 | 48 | 7 | 18 | 49 |
| P10S50D27 | 17 | 12 | 50 | 20 | 18 | 50 | 17 | 20 | 50 | 20 | 20 | 50 |

service routes, in most of the instances the mechanism finds a cost structure that yields an allocation in the core.

From Theorem 7, the inverse program $INVP$ is feasible. We found in our computational study that inverse problem together with single carrier rationality constraints is also feasible in all the instances. However, in some cases a feasible solution for $INVP+\{1\}+\{2\}$ could not be found. Note that for three carrier alliances a solution to $INVP+\{1\}+\{2\}$ means that a cost structure that yields payoffs in the core can be identified. For $INVP$ and $INVP+\{1\}$ we report if the feasible solution provided by CPLEX is in the core. For these cases there might be alternate solutions and some might provide an allocation in the core (for example, instances in which we find an allocation in the core by considering $INVP+\{1\}+\{2\}$ but not when we consider $INVP$ or $INVP+\{1\}$).

Table (6) suggests that a feasible solution to ($INVP$) in 10-25% of the instances directly yields an allocation in the core. As the inverse problem is constrained by adding single carrier rationality constraints in 25-45% of the instances the feasible solution yields an allocation in the core. Further $INVP+\{1\}+\{2\}$ is feasible in 70-95% of the cases, depending on the test class. Thus in 70-95% of the cases our mechanism provides an allocation in the core. Recall that it is not necessary that an instance will have a non-empty core.

### 5.5.2 Analysis of Different Test Classes

We analyze different test classes in Table 7. We consider alliances with three carriers. Ships and demand pairs are distributed uniformly among the carriers. We solve the ship assignment problem exactly and the utility of assigning a ship to a service route for a carrier is taken proportional to the sum of his flow times the revenue on the edges of the service route. First column in Table 7 denotes different problem classes. For each test class, the second column reports the average CPU time taken (averaged over

**Table 7:** Analysis of test classes.

| Test Class | Time | # Non -empty core | Non-empty core | | | Empty core | | |
|---|---|---|---|---|---|---|---|---|
| | | | %Unmet demand | Unused ships | %Utilizi -ation | %Unmet demand | Unused ships | %Utiliz -ation |
| P6S18D6 | 1.92 | 48 | 21.9 | 0.5 | 0.70 | 41.6 | 1 | 0.59 |
| P6S18D9 | 3.08 | 49 | 36.93 | 0.25 | 0.82 | 45.07 | 0 | 0.96 |
| P6S30D6 | 5.75 | 36 | 3.61 | 2.22 | 0.64 | 0.25 | 3.29 | 0.56 |
| P6S30D9 | 8.60 | 46 | 11.60 | 0.76 | 0.77 | 5.30 | 0.75 | 0.72 |
| P10S30D18 | 98.01 | 50 | 43.83 | 0.04 | 0.83 | N/A | N/A | N/A |
| P10S30D27 | 181.46 | 50 | 58.91 | 0 | 0.86 | N/A | N/A | N/A |
| P10S50D18 | 306.12 | 50 | 16.90 | 0.45 | 0.88 | N/A | N/A | N/A |
| P10S50D27 | 514.61 | 50 | 28.19 | 0 | 0.91 | N/A | N/A | N/A |
| P10S50D10 | 48.70 | 35 | 0.52 | 4.39 | 0.60 | 0.63 | 6.43 | 0.65 |

50 instances) in minutes to solve a problem instance. This includes the time to solve the service design problem for all the subsets of carriers and the time taken to solve the inverse problem. The third column represents the number of instances, out of a total of 50 random instances generated for each test class, for which an allocation in the core exists. To test if the core of a problem is non-empty, a linear program consisting of all the core inequalities is constructed and its feasibility is tested. The next three columns report the average percentage of unsatisfied demand to the total demand, the average number of un-utilized ships and the average utilization of capacity on the edges of the network respectively, for the instances with a non-empty core. The next three columns report same statistics for the cases with empty core.

The second column in Table 7 suggests that as the problem size (number of ports, ships or demand pairs) increases the time taken to solve the problem increases. Also, more than 95% of the time reported here is taken in solving the network design problem for various subset of carriers. The increase in time taken to solve the network design problem with the increase in problem size is similar to the trend reported in [1].

Note that among the test classes with 6 ports, P6S30D6 has the highest number of instances with an empty core. A closer look at Table 7 reveals that this test

class has the highest number of un-used ships and the lowest percentage of unsatisfied demand. Also instances in this class have lower utilization of capacity on the edges of the network. More specifically, these networks have over capacity. We constructed an additional class of instances namely P10S50D10. These instances also have over-capacity and show similar behavior as that of instances in the P6S30D6 class. Specifically, many instances in test class P10S50D10 also have empty core. This leads us to the conclusion that instances with over-capacity are more likely to have an empty core. The primary motivation for carriers to collaborate in liner shipping is that they do not have enough ships to maintain weekly frequency on the routes. For instances other than in P6S30D6 and P10S50D10 test class, since carriers and subset of carriers have few ships (as compared to the available demand), in most of the cases the grand alliance offers the best possibility for maintaining the required frequency on the service routes and thus most of the instances have non-empty core. Instances in P6S30D6 and P10S50D10 test classes, are however more likely to have profitable sub-coalitions. Our experiments yield that subsets of carriers that have good synergy in the origin- destination port of their demand triplets are more likely to form sub-coalitions. In general, if sub-coalitions have higher synergies (as compared to the grand alliance) then it is less likely that the grand alliance will be formed. Also there are fewer incentives for carriers to get into the organizational and managerial complexities of big alliances.

For 6 port instances with 18 ships, the percentage of unsatisfied demand is quite high. Further the average un-satisfied demand for the instances with empty core is even higher than the average un-satisfied demand for the instances with non-empty core. Thus instances with small fleet size in which carriers find synergies among themselves to satisfy higher demand are more likely to have a stable grand alliance. Also we note from Table 7 that as the size of the network increases from 6 to 10 ports all the instances in all the test classes (except P10S50D10) have non-empty core.

Instances with 10 ports that have very high demand as compared to the available fleet (un-satisfied demand is 40-60% of the total demand) are very likely to form stable grand alliance. In these instances, as there is a shortage of ships, only the grand alliance provides a global optimal schedule for the available fleet. Table 7 reflects that in fact the grand alliance schedules almost all the ships in the fleet and provides very high utilization of capacity (85-95%) on the operated routes.

Note that from Table 6 and Table7, if the core of a problem instance is non-empty, our mechanism succeeds in finding an allocation in the core in almost all (95-100%) of the instances, when the inverse problem is considered with all the subset rationality constraints.

### 5.5.3 Size and Number of Carriers

Next, we study the effect of number of carriers in an alliance. Table 8 reports results for alliances with two, three and four carriers. The first column represents the test class. To generate instances with $i$-carriers the number of demand pairs and ships are distributed uniformly randomly among $i$ carriers. Thus, Table 8 reports results for alliances among carriers with similar characteristics. The second and the third column report results for alliances with two carriers. The second column reports the number of instances (out of 50) for which an allocation in the core exists. The third column reports the average (average taken over 50 instances) percentage improvement in the total revenue generated by the alliance as compared to sum of the revenue generated by individual carriers working independently. The next two twins of columns report similar statistics for alliances with three and four carriers, respectively. For a particular instance, the percentage improvement in the revenue generated as a result of the alliance is computed as:

$$\frac{opt(N) - \sum_{i \in N} opt(\{i\})}{\sum_{i \in N} opt(\{i\})}.$$

**Table 8:** Analysis of the size of an alliance.

| Test Class | 2-Carriers | | 3-Carriers | | 4-Carriers | |
|---|---|---|---|---|---|---|
| | core | % improvement | core | % improvement | core | % improvement |
| P6S18D6 | 50 | 63.04 | 48 | 275.45 | 46 | 717.40 |
| P6S18D9 | 50 | 40.73 | 36 | 167.60 | 37 | 475.61 |
| P6S30D6 | 50 | 17.11 | 36 | 50.58 | 29 | 117.87 |
| P6S30D9 | 50 | 20.12 | 46 | 66.17 | 43 | 115.64 |
| P10S30D18 | 50 | 26.91 | 50 | 61.05 | 50 | 99.65 |
| P10S30D27 | 50 | 19.68 | 50 | 53.42 | 48 | 88.13 |
| P10S50D18 | 50 | 14.24 | 50 | 31.38 | 48 | 72.11 |
| P10S50D27 | 50 | 15.94 | 50 | 32.36 | 49 | 69.45 |

Table 8 suggests that as the number of carriers increases in an alliance, the alliance tends to become more un-stable in the sense that the number of instances with a non-empty core decreases. Note that as the number of carriers increases the number of constraints that need to be satisfied to obtain an allocation in the core increases exponentially. Higher the number of constraints higher the chances that the constraints will conflict with each other and thus higher the chances that the core will be empty. With higher number of carriers in an alliance it is more likely that some subset of carriers will have better synergies for an alliance than the grand alliance. Qualitatively, as the number of carriers increase in an alliance the organizational complexity of the alliance increases and the decision making process becomes time consuming. One of the most successful alliances among liner carriers have been the Maersk-Sealand alliance which consists of only two carriers [64]. Some of the bigger alliances have organized and re-organized themselves a number of times within a short span of time. For example, the Global Alliance which was formed in 1995 among four carriers (APL, OOCL, MOL and Nedlloyd) reorganized in 1998 to form the New World Alliance (NOL/APL, MOL, HMM) after the merger of APL and NOL in 1997.

The third, fifth and the seventh column of Table 8 clearly shows that alliances can generate higher revenues as compared to the carriers operating on their own. For

a particular instance, i.e. for a given set of ports, fleet size and demand pairs, as we increase the number of carriers (that is distribute the fleet and demand pairs among a larger number of carriers), the revenue that an individual carrier can generate reduces. However, the optimal solution of the grand alliance remains the same, independent of the number of carriers in the alliance. Thus, as reflected by Table 8 the the percentage increase in the revenue generated as a result of the alliance increases as the number of carriers increases.

### 5.5.4 Role and Contribution of Carriers in an Alliance

We study how participants with complementary and similar roles influence an alliance. Specifically, we study the alliance between a ship owner and a group of shippers. That is one player has all the ships and the other players have all the demand. First, we study instances (drawn from different test classes P6S18D6 - P10S50D27) with one ship owner and one shipper. This is a perfect situation for collaboration and all these instances have a non-empty core. Further, in all such instances our mechanism provides a cost structure such that the resulting payoffs to both the participants is in the core, when the inverse problem is solved with all the subset rationality constraints. Thus a stable alliance can be formed in all these instances. Next, we study problem instances with three shippers and a single ship owner. Depending on the problem instance ($P6S30D9$ etc) we found that the core is non-empty in 90%- 100% of the cases. Among the instances with non-empty core, in 95%-100% of the instances our mechanism provides a cost structure such that the resulting payoff to the participants is in the core. Comparing one ship owner and one shipper case with the one ship owner and three shippers case we conclude that in the latter case, shippers give rise to competition and instability in the grand alliance.

An interesting observation is that for $P6S30D6$ problem instances for one ship owner and three shippers case, 98% of the instances have non-empty core. Whereas,

for this test class when the ships and demand pairs are distributed uniformly among four carriers only 58% of the instances have non-empty core (Table 8). For the $P6S30D6$ class the number of ships are enough to satisfy most of the demand, thus even if there are three competing shippers the alliance is stable. As the ship owner has sufficient number of ships, he has an incentive to collaborate with as many shippers as possible to increase his revenue. Similarly, though the shippers compete for capacity on the ships, in the case when the system has over-capacity they can all form a sustainable alliance with the ship owner. However this is not the case when ships and demand pairs are uniformly distributed among four carriers as many subset of carriers find synergies to form sub-coalitions. In general, for other test classes also, instances with ships and demand pairs distributed among one ship owner and three shippers are more likely to have a non-empty core as compared to four equi-sized carriers. This is simply because in the former scenario the players have higher degree of complementarity in their role. Carriers have used conferences and alliances to fix price and moderate the buying power of shippers. As a result of our experiments we conclude that it is a good strategy for shippers also to form alliances and consolidate cargo before negotiating with the carriers and ship operators. This practice is observed in the industry as giant shippers and freight-forwarders consolidate the cargo of small shippers.

## *5.6 Conclusions*

In this chapter we addressed various problems posed by alliance formation among carriers in the liner shipping industry. We designed allocation mechanism for an alliance to share the benefits of the alliance in such a way that all the members are motivated to act in the best interest of the alliance. Since the revenue generated by the collaborative solution alone is not enough to guarantee the satisfaction of an individual carrier, the mechanism provides side payments to the carriers to motivate them to

"play along" in the alliance. Considering our preliminary results, we believe that the suggested solution approach has the potential to help carriers form sustainable alliances.

Our experiments suggest that across all test classes, in most (more than 95%) of the instances that have non-empty core our mechanism successfully finds a cost structure such that the resulting payoff to the carriers is in the core, when the inverse problem is solved with all the rationality constraints. Assignment of ships to the service routes influences the cost incurred by a carrier (thus his payoff) and the ownership of capacity on the edges by the carriers. However our results indicate that independent of the assignment of ships to the service routes in most of the cases our mechanism successfully finds a cost structure such that the overall payoff to the carriers is in the core. Analysis of different test classes suggests that the core is empty for a very high number of instances (more than 72%) drawn from the classes in which carriers have sufficient number of ships to satisfy the available demand. Further our experiments yield that, as the number of carriers increase in an alliance, the percentage improvement in the total revenue generated by the alliance as compared to the sum of the revenue generated by individual carrier independently increases. However, it becomes harder to find a solution in the core as the number of constraints to obtain a core allocation increase exponentially with the number of carriers. Further we conclude that carriers who have complementarity in their roles, for example ship owners and freight forwarders, are more likely to form stable alliances. Note that many other factors (such as compatibility in mission, strategy, governance, culture, organization and management etc of different partners of an alliance) also contribute significantly to the success of an alliance.

In this chapter we considered alliance formation among 3-4 carriers. For these alliances we considered all subset rationality constraints to find an allocation in the core. In many transportation and other logistics problems it is necessary to consider

alliances with higher number of participants. Also in liner shipping, smaller alliances are collaborating to from even bigger alliances, for example Grand Alliance and The New World Alliance laid down foundations for cooperation in 2006. However considering all rationality constraints becomes prohibitively expensive as the number of carriers increase in an alliance. To extend the mechanism developed in this chapter for alliances with higher number of participants, subset rationality constraints need to be added in the inverse problem in a constraint generation setting.

The liner shipping industry is deploying bigger and bigger ships. Further research is required to study the viability of bigger ships and their impact on alliances. These issues have also been studied by [35].

This chapter provides a basic framework for the research that can be used for designing allocation mechanism for various network design problems. The research integrates tools from optimization, economics and mathematics to study the selfish behavior of individual players in an alliance thus providing advances in interdisciplinary work.

# APPENDIX A

# LIST OF ABBREVIATIONS

## *A.1 Name of carriers*

APL : American President Line

CGM : Compagnie Generale Maritime

CMA : Compagnie Maritime d'Affrtement

COSCO : China Ocean Shipping Company

Hamburg Sud : Hamburg Sud Group

Hapag Lloyd : Hapag-Lloyd AG

HMM : Hyundai Merchant Marine

K Line : Kawasaki Kisen Kaisha

Maersk : A.P.Moller-Maersk Line

Marfret : Marfret Compagnie Maritime

MISC : Malaysian International Shipping Corporation

MOL : Mitsui OSK Lines

NYK : Nippon Yusen Kaisha

OOCL : Orient Overseas Container Lines

P&O : P & O Container Line

Sea-Land : Sea-Land Service Inc.

Yang Ming : Yang Ming Line

# REFERENCES


[1] Agarwal, R. and Ergun, O., "Ship scheduling and network design for cargo routing in liner shipping," *to appear in Transportation Science*, 2007.

[2] Agarwal, R., Houghtalen, L., Ergun, O., and Ozener, O., "Collaboration in cargo routing." Working paper, 2007.

[3] Ahuja, R. K. and Orlin, J. B., "Inverse optimization," *Operations Research*, vol. 49, no. 5, pp. 771–783, 2001.

[4] Ahuja, R. K., Orlin, J. B., and Sharma, D., "A composite very large scale neighborhood structure for the capacitated minimum spanning tree problem," *Operations Research Letters*, vol. 31, pp. 185–194, 2003.

[5] American Association of Port Authorities, "America's ports today," Feb. 2006. AAPA Policy Paper.

[6] APL. www.apl.com, Accessed: 2005.

[7] Barnhart, C., Johnson, E. L., Anbil, R., and Hatay, L., "A column-generation technique for the long-haul crew-assignment problem," in *Optimization in industry 2: Mathematical programming and modeling techniques in practice*, pp. 7–24, John Wiley & Sons, Inc., 1994.

[8] Barry Rogliano Salles- AlphaLiner, "Liner shipping report," January 2006. Archived Report.

[9] Bendall, H. B. and Stent, A. F., "Longhaul feeder service in an era of changing technology : an asia-pacific perspective," *Maritime Policy and Management*, vol. 26, no. 2, pp. 145–159, 1999.

[10] Benders, J. F., "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[11] Bertsimas, D. and Tsitsiklis, J. N. in *Introduction to Linear Optimization*, Athena Scientific, 1997.

[12] Bondareva, O., "The core of an n-person game," *Vestnik Leningrad Univ.*, vol. 13, pp. 141–142, 1962.

[13] Chandrashekar, T. S. and Narahari, Y., "Procurement network formation: A cooperative game approach." Working paper, 2006.

[14] Cheung, R. K. and Chen, C.-Y., "A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem," *Transportation Science*, vol. 32, no. 2, pp. 142–162, 1998.

[15] Christiansen, M., Fagerholt, K., and Ronen, D., "Ship routing and scheduling: Status and perspectives," *Transportation Science*, vol. 38, no. 1, pp. 1–18, 2004.

[16] Christiansen, M. and Nygreen, B., "A method for solving ship routing problems with inventory constraints," *Annals of Operations Research*, vol. 81, pp. 357–378, 1998.

[17] Cordeau, J.-F., Soumis, F., and Desrosiers, J., "A benders decomposition approach for the locomotive and car assignment problem," *Transportation Science*, vol. 34, pp. 133–149, 2000.

[18] Cordeau, J.-F., Soumis, F., and Desrosiers, J., "Simultaneous assignment of locomotives and cars to passenger trains," *Operations Research*, vol. 49, pp. 531–548, 2001.

[19] Cordeau, J.-F., Stojkovic, G., Soumis, F., and Desrosiers, J., "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation Science*, vol. 35, pp. 375–388, 2001.

[20] Dembo, R. and Rosen, D., "The practice of portfolio replication: A practical overview of forward and inverse problems," *Annals of Operations Research*, vol. 85, no. 0, pp. 267–284, 1999.

[21] Derks, J. J. M. and Tijs, S. H., "Stable outcomes for multicommodity flow games," *Methods of Operations Research*, vol. 50, pp. 493–504, 1985.

[22] Dial, R., "Minimal revenue congestion pricing. part ii:an efficient algorithm for the general case," *Transportation Research Part B*, vol. 34, pp. 645–665, 2000.

[23] Drewry, "Global container markets," *Drewry Shipping Consultants Limited*, July 1996.

[24] Drewry, "The drewry container market quarterly," *Drewry Shipping Consultants Limited*, vol. September 2001, September 2001.

[25] Dror, M., "The traveling salesman, binpacking and the knapsack," *Applied Mathematics and Computation*, vol. 35, pp. 195–207, 1990.

[26] Engevall, S., Gothe-Lundgren, M., and Varbrand, P., "The heterogeneous vehicle-routing game," *Transportation Science*, vol. 38, no. 1, pp. 71–85, 2004.

[27] Fagerholt, K., "Optimal fleet design in a ship routing problem," *International Transactions in Operational Research*, vol. 6, pp. 453–464, 1999.

[28] Florian, M., Bushell, G., Ferland, J., Guerin, G., and Nastansky, L., "The engine scheduling problem in a railway network," *INFOR*, vol. 14, pp. 121–138, 1976.

[29] Garey, M. R. and Johnson, D. S., "A list of np-complete problems," in *Computers and Intractability: A Guide to the Theory of NP-Completeness*, pp. 214–215, W. H. Freedman and Co., 1979.

[30] Goemans, M. and Skutella, M., "Cooperative facility location games," *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[31] Granot, D., Hamers, H., and Tijs, S., "On some balanced, totally balanced and submodular delivery games," *Mathematical Programming*, vol. 86, no. 2, pp. 355–366, 1999.

[32] Hamers, H., "On the concavity of delivery games," *European Journal of Operational Research*, vol. 99, no. 2, pp. 445–458, 1997.

[33] Hershberger, J. and Suri, S., "Vickery prices and shortest paths: What is an edge worth?," in *Annual IEEE Symposium on Foundation of Computer Science*, 2001.

[34] Houghtalen, L., Ergun, O., and Sokol, J., "Designing allocation mechanisms for carrier alliances." Working paper, 2006.

[35] Imai, A., Nishimura, E., Papadimitriou, S., and Liu, M., "The economic viability of container mega-ships," *Transportation Research Part E*, vol. 42, pp. 21–41, 2006.

[36] Jain, K. and Vazirani, V. V., "Applications of approximation algorithms to cooperative games," *In Proceedings of 33rd ACM Symposium on Theory of Computing*, pp. 364–372, 2001.

[37] Kalai, E. and Zemel, E., "Generalized network problems yielding totally balanced games," *Operations Research*, vol. 30, no. 5, pp. 998–1008, 1981.

[38] Kalai, E. and Zemel, E., "Totally balanced games and games of flow," *Mathematics of Operations Research*, vol. 7, no. 3, pp. 476–478, 1982.

[39] Kubo, M. and Kasugai, H., "On the core of the network design game," *Journal of the Operations Research Society of Japan*, vol. 35, pp. 250–255, 1992.

[40] Lin, S. and Kernighan, B., "An effective heuristic algorithm for the travelling salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

[41] M. Gothe-Lundgren, K. J. and Varbrand, P., "on the nucleolus of the basic vehicle routing game," *Mathematical Programming*, vol. 72, pp. 83–100, 1996.

[42] Markakis, E. and Saberi, A., "On the core of the multicommodity flow game," in *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pp. 93–97, ACM Press, New York, USA, 2003.

[43] McDaniel, D. and Devine, M., "A modified bender's partitioning algorithm for mixed integer programming," *Management Science*, vol. 24, pp. 312–379, 1977.

[44] Midoro, R. and Pitto, A., "A critical evaluation of strategic alliances in liner shipping," *Maritime Policy and Management*, vol. 27, no. 1, pp. 31–40, 2000.

[45] N. Hingorani, D. M. and Tornqvist, K., "Setting a new course in the container shipping industry," *IBM Business Consulting Services*, vol. Travel and Transportation, 2005.

[46] Nisan, N. and Ronen, A., "Algorithmic mechanism design," *Games and Economic Behavior*, vol. 35, pp. 166–196, 2001.

[47] OOCL. www.oocl.com, Accessed: 2005.

[48] Owen, G., "On the core of linear production games," *Mathematical Programming*, vol. 9, pp. 358–370, 1975.

[49] Perakis, A. N., "Fleet operations optimization and fleet deployment," in *The Handbook of Maritime Economics and Business* (Grammenos, C. T., ed.), pp. 580–597, Lloyd's of London, 2002.

[50] Rana, K. and Vickson, R. G., "Routing container ships using lagrangean relaxation and decomposition," *Transportation Science*, vol. 25, no. 3, pp. 201–214, 1991.

[51] ROI, "Profit optimization for container carriers," July 2002. White Paper.

[52] Ronen, D., "Cargo ships routing and scheduling: Survey of models and problems," *European Journal of Operational Research*, vol. 12, pp. 119–126, 1983.

[53] Ronen, D., "Ship scheduling: The last decade," *European Journal of Operational Research*, vol. 71, no. 3, pp. 325–333, 1993.

[54] Roughgarden, T. and Tardos, E., "How bad is selfish routing," *Journal of the ACM*, vol. 49, no. 2, pp. 236–259, 2002.

[55] Ryoo, D. K. and Thanopoulou, H. A., "Liner alliances in the globalization era: A strategic tool for Asian container carriers," *Maritime Policy and Management*, vol. 26, no. 4, pp. 349–367, 1999.

[56] Samet, D. and Zemel, E., "On the core and dual set of linear-programming games," *Mathematics of Operations Research*, vol. 9, no. 2, pp. 309–316, 1984.

[57] Sánchez-Soriano, J., "Pairwise solutions and the core of transportation situations," *European Journal of Operational Research*, vol. 175, pp. 101–110, 2006.

[58] Savelsbergh, M. W. P., "A branch-and-price algorithm for the generalized assignment problem," *Operations Research*, vol. 45, pp. 831–841, 1997.

[59] Shapley, L., "On balanced sets and cores," *Naval Research Logistics Quarterly*, vol. 14, pp. 453–460, 1967.

[60] Shapley, L., "Cores of convex games," *International Journal of Game Theory*, vol. 1, pp. 11–26, 1971.

[61] Shapley, L. and Shubik, M., "The assignment game i: The core," *International Journal of Game Theory*, vol. 1, pp. 111–130, 1972.

[62] Shen, W. S. and Khoong, C. M., "A dss for empty container distribution planning," *Decision Support Systems*, vol. 15, pp. 75–82, 1995.

[63] Slack, B., Comtois, C., and McCalla, R., "Strategic alliances in the container shipping industry: A global perspective," *Maritime Policy and Management*, vol. 29, no. 1, pp. 65–76, 2002.

[64] Song, D. W. and Panayides, P. M., "A conceptual application of cooperative game theory to liner shipping strategic alliances," *Maritime Policy and Management*, vol. 29, no. 3, pp. 285–301, 2002.

[65] United States Customs Service, "Singapore, the World's busiest seaport, implements the container security initiative and begins to target and pre-screen cargo destined for U.S.," March 2003. Archived Press Release.

[66] Wiel, R. J. V. and Sahinidis, N. V., "An exact solution approach for the time dependent travelling salesman problem," *Naval Research Logistics*, vol. 43, pp. 797–820, 1996.

[67] Yen, K. P., "Strategic alliances in the linear shipping industry," Master's thesis, Massachusetts Institute of Technology, 1994.

# VITA

Richa Agarwal received the Masters of Science (Integrated) degree in Mathematics and Scientific Computing from the Indian Institute of Technology, Kanpur, India in 2000. She received her Masters of Science degree in Industrial and Systems Engineering from the University of Florida, Gainesville in 2002 where she worked with Prof. R. K. Ahuja on developing large scale neighborhood search algorithms for the Vehicle Routing Problem. She joined the Ph.D. program in Algorithms, Combinatorics and Optimization program in the School of Industrial and Systems Engineering at the Georgia Institute of Technology in 2002 where she worked with Dr. Özlem Ergun on the Parallel Machine Scheduling Problem and various problems in Liner Shipping. This thesis covers some of this latter work.

Richa's research interests are in mathematical programming, large scale optimization and algorithmic game theory. She is also interested in studying various logistics and scheduling problems. In 2006, she received the Georgia Institute of Technology student paper award sponsored by the SAIC and the NSF student travel grant sponsored by the NSF and the Arizona State University. Richa is a member of INFORMS and is a student representative on the INFORMS chapter/fora committee.