NTNU – Trondheim
Norwegian University of
Science and Technology

The Liner Shipping Network Design Problem

Strengthened formulations considering
complex route structures, transhipment and
transit time

Marie Ameln
Julie Sand Fuglum

Industrial Economics and Technology Management
Submission date: June 2015
Supervisor: Henrik Andersson, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem description

The purpose of this thesis is to develop new extensive mathematical formulations of the Liner Shipping Network Design Problem (LS-NDP) and to use theory and methods to strengthen the formulations. The aim of the work will be on making models that reflect a realistic situation and therefore transhipment, the cost of transhipment, a heterogeneous fleet, route dependent capacities, complex route structures and transit time will be taken into account. Since the LS-NDP is such a complex problem that cannot be solved in polynomial time there will also be a focus on strengthening the formulations through symmetry breaking constraints and valid inequalities.

Preface

This master thesis is the last part of our Master of Science degree at the Norwegian University of Science and Technology (NTNU), Department of Industrial Economics and Technology Management. The thesis was written during the spring semester of 2015. It is done within the field of operations research.

We present new comprehensive exact formulations of the *Liner Shipping Network Design Problem* (LS-NDP) with a particular focus on enabling complex route structures, transhipment and transit time.

We would like to thank our supervisors Magnus Stålhane, Kristian Thun and Henrik Andersson for their sincere enthusiasm about our work, valuable feedback and guidance. We will surely miss our weekly discussions.

We also want to thank Berit D. Brouer, who has been an inspiration for our work, and we had the pleasure to discuss the LS-NDP with at the NTNU Ocean week conference in Trondheim.

At last we want to convey that working with the Liner Shipping Network Design Problem has been extremely interesting. As we have dealt with this problem now for some time its complex nature reveals more and more, and we have understood that there is no simple solution to it. However, that is also why it has been so exciting to work within this field. We are convinced that the liner shipping industry has a lot to gain from operations research.

Trondheim, June 11th, 2015

Marie Ameln   Julie Sand Fuglum

Marie Ameln   Julie Sand Fuglum

Sammendrag

Internasjonal container linjefart er en milliardindustri, som opplever stadig lavere marginer. Likevel har det vært begrenset forskning på planleggingsproblemer innen bransjen. Arbeidet som er presentert i denne rapporten er motivert av en overbevisning om at det er et stort potensiale for kostnadseffektivisering innen linjefart, ved å forbedre ruteplanleggingsprosessene.

Det har blitt utviklet tre matematiske blandede heltallsmodeller for å løse linjefarts nettverk-problemet, også kalt the Liner Shipping Network Design Problem (LS-NDP). Modellene blir også utvidet til å ta hensyn til krav om maksimal transittid på lastene, noe som gjør at det totalt blir presentert seks formuleringer. I tillegg til hvorvidt formuleringene tar hensyn tid transittid eller ikke, skiller de seg fra hverandre med tanke på hvilke rutestrukturer de muliggjør.

Målet for alle modellene er å minimere kostnadene til et shippingselskap ved å utforme et effektivt rutenettverk, som tilfredsstiller alle etterspørselskrav. Alle formuleringene inkluderer vesentlige funksjoner som omlasting, omlastingskostnader, et ukentlig frekvenskrav og en heterogen skipsflåte.

Det største fokusområdet i arbeidet har vært å utvikle formuleringer som åpner for mer fleksible rutestrukturer. Den første formuleringen, *the Basic model*, tillater kun enkle rutestrukturer. Dette vil si at en havn kun kan besøkes én gang av en gitt rute i en enkelt rotasjon. De to andre modellene muliggjør forskjellige typer komplekse rutekonstruksjoner. *The Flower model* tillater ruter hvor én port kan besøkes en rekke ganger av samme rute. *The Chain model* muliggjør rutestrukturer hvor en flere havner kan besøkes to ganger av samme rute. Vi har ikke gjort noen kompromisser som følge av de komplekse rutestrukturene, og alle omlastingskostnadene er riktig beregnet.

Utifra litteraturstudien er vårt inntrykk at i de mest omfattende LS-NDP modellene presentert så langt, er de komplekse rutene begrenset til å kun kunne besøke én port maks to ganger, en såkalt *butterfly route*. Så vidt vi vet, er dette første gang eksakte løsninger av modeller med mer komplekse rutestrukturer blir presentert. Vi har studert rutenettverket til flere store shippingselskaper, og funnet at et betydelig antall av disse er komplekse. Dette har gitt oss tro på at dette arbeidet kan ha praktisk relevans.

Modellene presentert i denne rapporten er av høy beregningsmessig kompleksitet, noe som gjør at de tar lang tid å løse. På grunn av dette har vi utviklet symmetribrytende restriksjoner og gyldige ulikheter som forsøk på å styrke formuleringene. Testingen av flere av disse gir gode resultater, og indikerer at den generelle løsningstiden forbedres. Disse ble inkludert i videre testing av modellene.

Instansene løst til optimalitet er sammenlignbare i størrelse med instansene i eksisterende litteratur om eksakte metoder for LS-NDP. Resultatene viser, ikke overraskende, at de komplekse modellene tas vesentlig lenger tid å løse til optimalitet.

Resultatene fra studien viser fordelen med å muliggjøre komplekse rutestrukturer, og hvordan dette bidrar til en bedre allokering av skipsflåten. En interessant observasjon fra studien av transittid-modellene er at det virker som fordelen av å tillate komplekse rutestrukturer blir forsterket når det er strengere krav til transittid.

Abstract

International liner container shipping is a multibillon dollar business, experiencing increasingly pushed margins. However, there have been scarce research efforts when it comes to planning and scheduling problems within this industry. The work done in this report is motivated by the conviction that shipping companies may expect large benefits from improving the routing processes of their ships.

Three extensive mathematical models for the liner shipping network design problem (LS-NDP) have been developed. The models have been extended to also take into account transit time, making it six formulations in total. Besides whether they incorporate transit time, the models differ in which route structures they enable.

We take the perspective of a liner shipping company. The objective of all the models is to minimize costs by designing an efficient network of routes that satisfy all demand. All the formulations include important features as transhipment, transhipment cost, a weekly frequency requirement and a heterogeneous fleet.

The key focus area has been on developing formulations that allow for more flexible route structures. The first formulation, called *the Basic model*, only allows simple route networks. The two other models enable different types of complex route structures. *The Flower model* allows the routes in the network to visit one port in the route multiple times, while *the Chain model* incorporates the possibility of a route visiting several ports in the route twice. No compromises have been made in allowing the complex route structures, and all transhipment costs are calculated correctly.

To the best of our knowledge the most comprehensive liner shipping models in literature have restricted the routes to have at most one port that is visited at most twice, socalled *butterfly routes*. As far as we know, this is the first time exact solutions allowing even more complex route structures, have been presented. Studying the route-nets of the world leading liners and finding that a substantial number of the routes they provide are complex routes, have given us a belief that there is a practical relevance of this research.

Since the models are of non-trivial complexity, a strong effort has been made in order to

strengthen the formulations to make them more computationally efficient. This has been done by imposing symmetry breaking constraints and valid inequalities. The testing of the strengthening formulations show good results for several of them, and they are included in the models in further computational study.

The instances solved to optimality are comparable in size to the instances in existing literature of exact methods considering the LS-NDP. The computational results show that the complex models take significantly longer time to solve to optimality.

The results show the advantage of more complex route structures and how it enables a better allocation of the vessels. The results from the computational study of the transit time models indicate that the advantage of allowing more flexible route structures, is amplified as the transit time requirements get stricter.

Contents

List of Tables

List of Figures

Glossary

l

LS-NDP	Liner Shipping Network Design Problem
NM	Nautic Mile
TEU	The twenty-foot equivalent
FEU	The forty-foot equivalent unit
RORO	Roll-on/roll-off ships are vessels designed to carry wheeled cargo
Dwt	Deadweight tonnage
GDP	Gross domestic product
Trade route	Trade between an origin and destination group of countries
Butterfly route	A route where one port is visited twice by the same route
Flower route	A route that can visit one and only one port in the route multiple times
Chain route	A route where multiple ports can be visited at most twice in the same route
Butterhub	A port that is visited multiple times in one route.
Butterarc	An arc that is used multiple times in one route.
Transhipment	Cargo that is unloaded in a hub and picked up by another vessel.
Internal transhipment	Transhipment within the same route.
Exernal transhipment	Transhipment among different routes.

Chapter 1

Introduction

An efficient liner shipping network is of great importance both economically and environmentally. The process of designing the service routes of a liner shipping company thus is essential for the competitiveness of the company and its ability to sustain its share of the global containerized freight market. The problem of determining the structure of the route network is often referred to as the liner shipping network design problem (LS-NDP). The LS-NDP is NP-hard (Agarwal and Ergun, 2010). In this report six comprehensive mixed integer programming (MIP) models for the LS-NDP are presented. This initial chapter provides an introduction to the problem as well as the motivation, purpose and focus of our work. At the end of the chapter the content and organization of the report are presented.

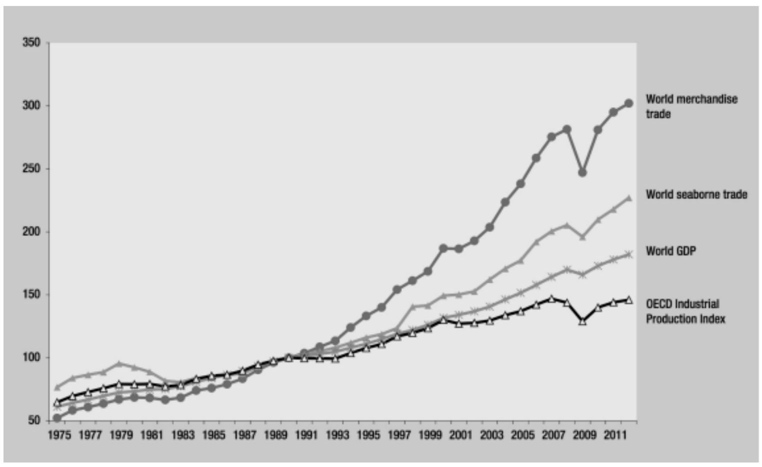


Figure 1.1: The OECD Industrial Production Index and indices for world GDP, world merchandise trade and world seaborne trade (1975-2012) (1990 = 100). (Asariotis et al., 2014).

Motivation. There are particularly four reasons to why there is a great potential of using operations research (OR) in the liner shipping industry:

1. **Increased world seaborne trade.** Economic globalization and increased world trade have led to rapid growth in the liner shipping industry for the last decades. The number of containers shipped increased at an average rate of 8.2% per year between 1990 and 2010, and World Shipping Council estimates that 60% of all goods by value are at one point transported by container ships (World Shipping Council, 2014b). This is more than US $4 trillion worth of goods annually. Figure 1.1 shows that the world seaborne trade has been growing relatively faster than the world gross domestic product (GDP). A consequence of the increase in transportation is greater emissions, and the liner shipping industry makes up nearly 3 % of the world's total discharge of CO2. In a market as large as the liner shipping market even a small cost reduction will have great effect both economically and environmentally.

2. **Lower margins.** In the last years, when the world economy has seen little growth following the financial crisis in 2008, the total supply in the shipping market has increased by 37 %, far more than the worldwide economic growth in the same period, see Figure 1.1. The oversupply in the freight market leads to lower freight rates, pushing the margins of the shipping companies (Asariotis et al., 2014). Because of this the shipping companies are searching for new potential cost reductions.

3. **Complexity.** The LS-NDP is complex by nature, and it is difficult to capture all the operating features of liner shipping. The composite traits of the problem make it impossible for the liner shipping companies to make optimal decisions based on their traditional planning approaches. The container paths are actually designed manually by experienced managers. Evidently, manual planning is inefficient and may not find all possible container paths (Wang et al., 2013). This demonstrates the great potential of operations research.

4. **Few research efforts on LS-NDP.** The liner container shipping industry is more conservative than other transportation modes, such as the airline industry (Meng et al., 2014). The global shipping companies have traditionally been sensitive about their operating data, and been reluctant to share their data and concerns with researchers. Recently, however, several leading container shipping companies have sought to use OR methods to make better decisions because of the increased competition and high bunker prices in the container shipping industry. Only a few research efforts have actually been implemented by liner shipping companies, and existing reviews on ship routing and scheduling have mainly focused on industrial and tramp shipping. Therefore there is clearly a significant

opportunity for academic research to address practical containership routing and scheduling problems. This is suggested as an area for future research by among others (Meng et al., 2014).

Purpose. Kjeldsen (2011) points out that literature related to liner shipping is largely concerned with specific applications which is why the problem formulations and solution methods are much diversified. The purpose of the work presented in this report is to develop new and more flexible formulations of the Liner Shipping Network Design Problem (LS-NDP). Our aim is to include more of the important operating features of the problem in order to represent the model more realistically - and then again to obtain better solutions. As emphasized above an improved route network can make a great difference for a liner company. One of the most challenging aspects of the LS-NDP is the modelling of complex route structures, i.e. routes where one or several ports are visited more than once. Since complex routes often represent a major part of the total route schedule provided by a liner company - the rich formulations enabling complex routes are of great importance.

Figure 1.2 shows Maersk's Asia-Europe Roundtrip. In this roundtrip Rotterdam and several other cities are visited twice. Thus this illustrates the practical relevance of enabling complex routes in the model. To the best of our knowledge previous work on complex route structures has been scarce and characterized by a high level of simplification. In this report we therefore have a great focus on the modelling of complex route structures.

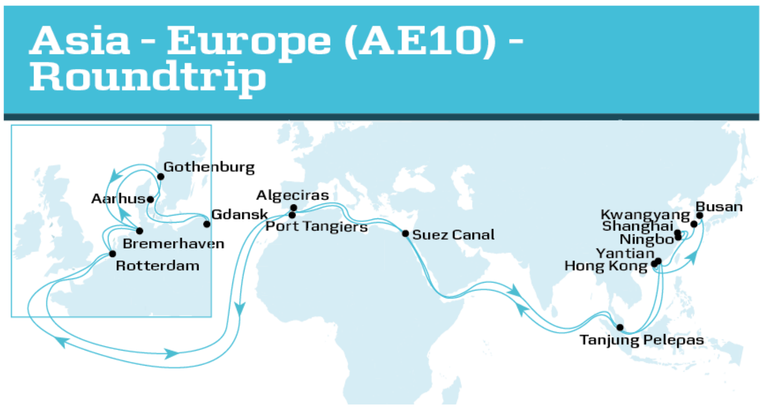
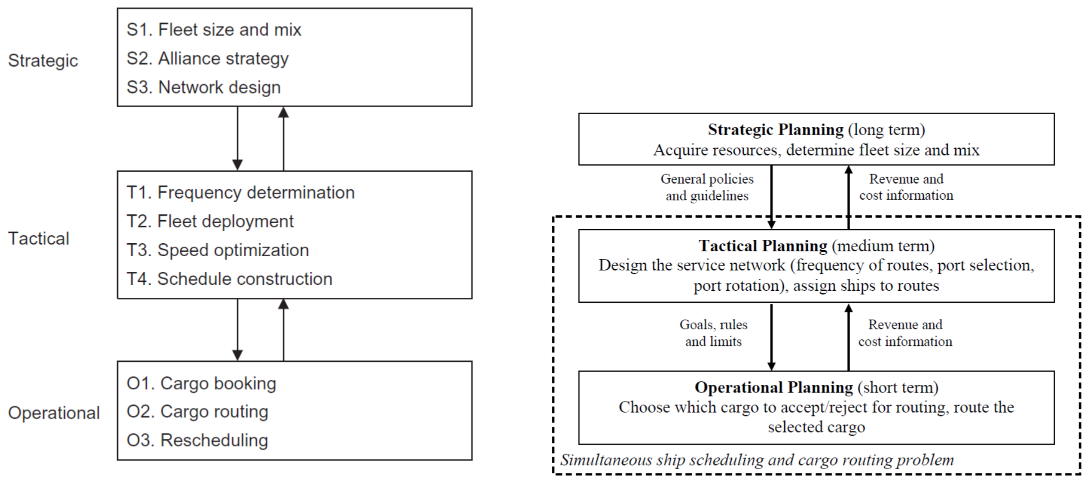


Figure 1.2: In Maersk's Asia-Europe-Roundtrip several ports are visited twice (Maersk, 2014a). This illustrates the practical relevance of allowing complex route structures in the model.

Level of decision making. Decision problems are often categorized as operational, tactical or strategic, based on the timeline of planning horizon for the decisions that need to be taken. For global liner shipping companies it is common to publish their routenet on a yearly basis. Whether the network design problem is regarded as a strategic or tactical problem differ among papers. Meng et al. (2014) categorize the LS-NDP as a strategic decision problem. Agarwal and Ergun (2010), on the other hand, defines it as a tactical problem, see Figure 1.3b. How the two papers view the levels of decision making for liner shipping companies is depicted in Figure 1.3. This may indicate that LS-NDP is on the border between belonging to the strategic or tactical category.


Strategic
S1. Fleet size and mix
S2. Alliance strategy
S3. Network design
Tactical
T1. Frequency determination
T2. Fleet deployment
T3. Speed optimization
T4. Schedule construction
Operational
O1. Cargo booking
O2. Cargo routing
O3. Rescheduling
Strategic Planning (long term)
Acquire resources, determine fleet size and mix
General policies and guidelines
Revenue and cost information
Tactical Planning (medium term)
Design the service network (frequency of routes, port selection, port rotation), assign ships to routes
Goals, rules and limits
Revenue and cost information
Operational Planning (short term)
Choose which cargo to accept/reject for routing, route the selected cargo
Simultaneous ship scheduling and cargo routing problem

(a) Meng et al. (2014) categorize network design as a strategic problem

(b) Agarwal and Ergun (2008) categorize network design as a tactical problem

Figure 1.3: Levels of decision making in liner shipping. Whether the network design problem is regarded as a strategic or tactical problem differ among papers.

Priorities and trade-offs. The terms *reality representation* and *solution speed* are classical trade-offs in OR-modeling, and refer to the simple fact that an “increase” in a model’s reality representation will almost always lead to a decrease in solution speed. As both normally are seen to be positive from a users’ point of view, they introduce a difficult trade-off in practical modeling (Nygreen et al., 1998). Considering the fact that this usually is a problem that only needs to be solved at most every year, we have given the solution speed lower priority than the reality representation. However, we have attempted to find other ways of reducing the solution speed, and made a great effort in finding and testing different symmetry breaking constraints and valid inequalities to strengthen the formulations.

Content and organization. The report is organized as follows; In Chapter 2 we present the liner container shipping industry, where we look into the world of liner shipping and containerized goods. We then review relevant literature in Chapter 3. In this chapter we look into important aspects in liner shipping and how these are handled in different papers, and also different network designing problems and how these relate to the LS-NDP. This is followed by our problem description in Chapter 4.

The mathematical formulations are presented in Chapters 5 to 8. First, we present the Basic model in Chapter 5, a formulation that only allows simple routes. In Chapter 6 and Chapter 7 we present two models that allow for more complex route structures, the *Flower* model and the *Chain model*, respectively. In Chapter 8 the three models are extended in order to incorporate transit time.

In Chapter 9 we propose different symmetry breaking constraints and valid inequalities in order to strengthen the formulations presented in Chapter 5-7. In Chapter 10 we describe the instances used in the computational study. The strengthening formulations are tested and evaluated in Chapter 11. Technical and economical results from a computational study are discussed in Chapter 12, before we conclude and make suggestions for further research in Chapter 13.

Chapter 2

Liner container shipping

There are generally three modes of operations in shipping: industrial, tramp, and liner (Lawrence, 1972). Industrial shipping is characterized by cargo owner also owning the ship. Tramp ships are like taxis as they follow the available cargo and liners, see Figure 2.1 are operated somewhat like bus lines with a published itinerary and schedule (Christiansen et al., 2004). *Liner shipping routes* are characterized by cyclic routes repeatedly sailed during a scheduled horizon and transhipment of cargo in hub ports. Liner ships have huge capacities in which makes one journey very efficient.

Figure 2.1: Liner container ship

The liner shipping industry has contributed to advances in the standard of living of most of the world's population in the last 35 years, as the gains from trade through advancing global commerce were enabled by a reliable, efficient and relatively low-cost transportation provided by the industry (Insight, 2009).

According to the World Shipping Council (WSC) [1] there are approximately 400 liner services in operation today, most providing weekly departures from all the ports that each service calls. Liner shipping companies publish their service routes, with fixed sequences of ports of call at a regular service frequency, to attract cargo. A liner ship service route may not have an origin or destination as the voyages become closed routes (Rönen, 1983).

Liner shipping mainly involves transporting containerized cargo on the regularly serviced routes. In this we consider liner shipping with *containerized cargo*, and this industry will be described in the following sections.

2.1 Containerized goods

Maersk, one of the world's largest overseas cargo carriers, writes about containers on their webpages; "Over the past 60 years this unremarkable metal box has overhauled global industry and transformed world trade, kickstarting the modern age of consumerism" (Maersk, 2014b). Figure 2.3 shows how the growth in containerized trade has been the last 18 years. Today, more than 50% of the value of global seaborne trade is transported in containers, see Figure 2.2.

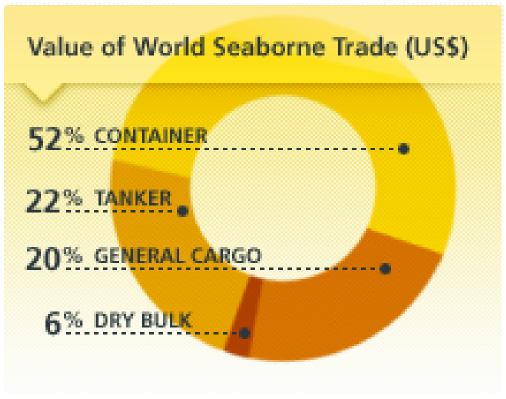


Figure 2.2: Half of the value of world seaborne trade is transported in containers (World Shipping Council, 2014a).

Container shipping uses containers of various sizes to load, transport, and unload goods. The container sizes are standardized so that the containers can be most efficiently stacked and so that ships, trains, trucks and cranes at the ports can be specially fitted or built to a single size specification. The two most important, and most commonly used sizes today, are the 20-foot (TEU) and 40-foot (FEU) lengths. The 20-foot container is considered the industry standard reference so cargo volume and vessel capacity are commonly measured in TEU (World Shipping Council, 2014a).

[1]The World Shipping Council is the peak industry trade group representing the international liner shipping industry

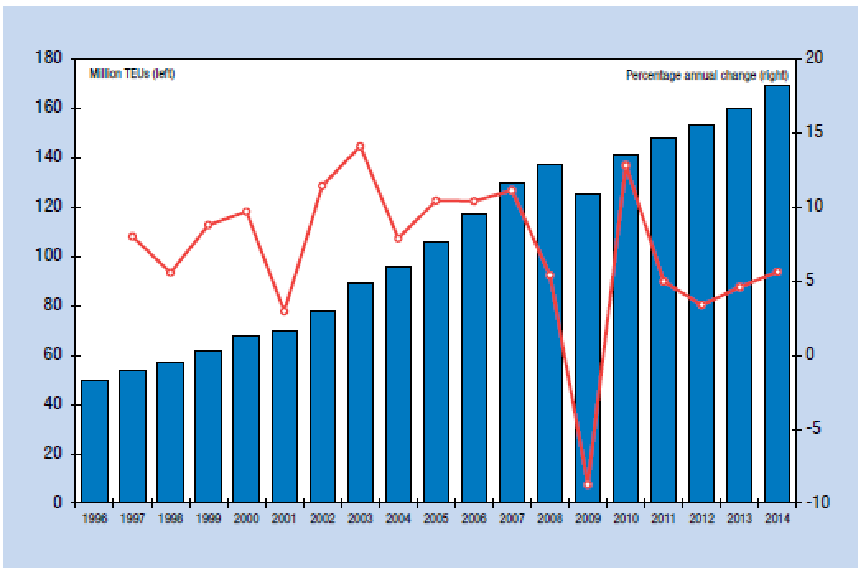


Figure 2.3: Global containerized trade, 1996–2014 (Millions of TEUs and percentage annual change). (Asariotis et al., 2014)

History of containerization

Modern container shipping has existed since 1961, when the international standards for container size was agreed on. Since then this method of transport for goods has grown steadily.

However, according to the World Shipping Council, the idea of using some type of shipping container was built on old concepts. Boxes similar to modern containers were used for combined rail- and horse-drawn transport in England since 1792. The US government used small standard-sized containers during the Second World War, which proved a means of efficiently unloading and distributing supplies. In 1955, Malcom P. McLean, a trucking entrepreneur from North Carolina, USA, bought a steamship company with the idea of transporting entire truck trailers with their cargo still inside. He realized it would be much simpler and quicker to have one container that could be lifted from a vehicle directly on to a ship without first having to unload its contents (World Shipping Council, 2014a).

McLean's ideas were based on the theory that efficiency could be vastly improved through a system of *intermodalism*, in which the same container, with the same cargo, can be transported with minimum interruption via different transport modes during its journey. Containers could be moved seamlessly between ships, trucks and trains. This would simplify the whole logistical process and, eventually, implementing this idea led to a revolution in cargo transportation and international trade over the next 60 years (World Shipping Council, 2014a).

The value chain

Goods transported by ocean containers on liner ships can be placed into the container at the factory origin. The container is locked and sealed so the goods can be safely secured inside the container until it arrives at the purchaser's warehouse, factory or store.

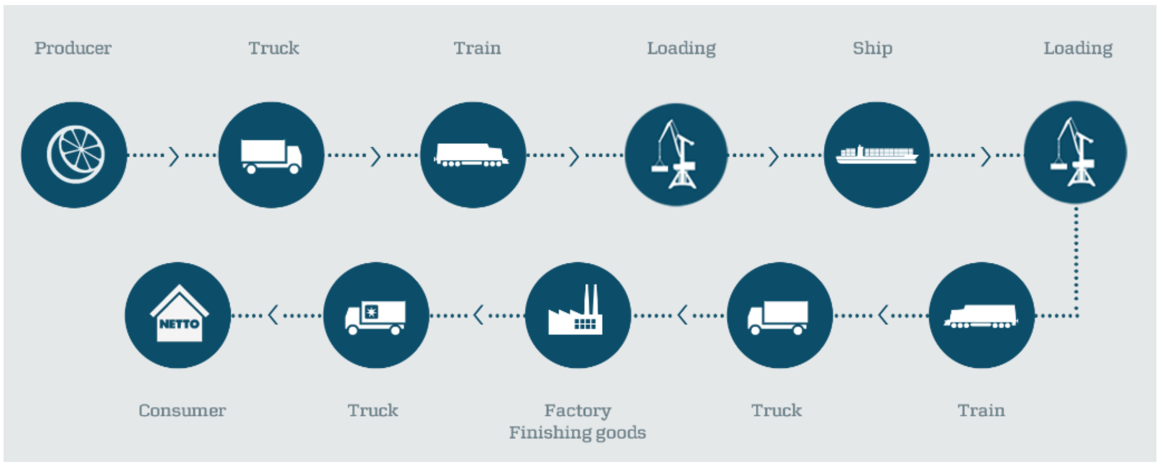


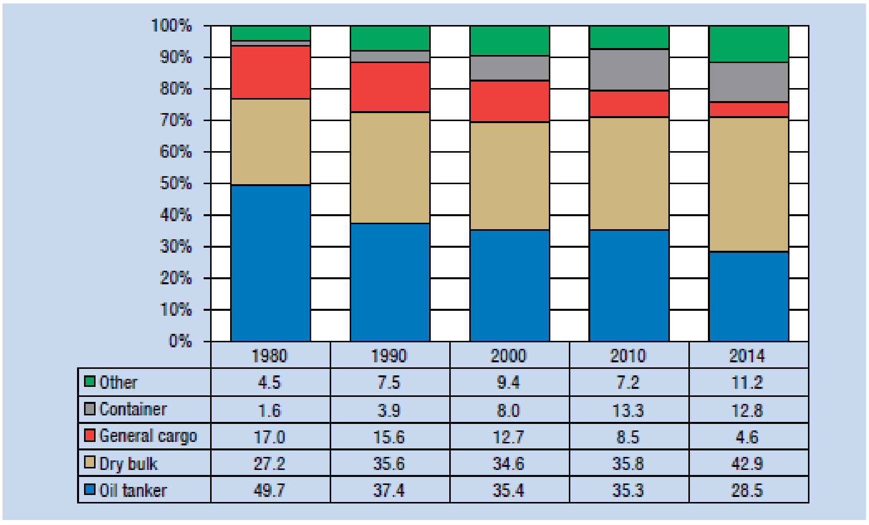
Figure 2.4: The value chain of containerized goods in liner shipping. The global supply chains connect producers on one side of the planet to consumers on the other (Asariotis et al., 2014).

The value chain of containerized goods is illustrated in Figure 2.4. An example of a process can be described as follows: A store places an order of a product manufactured far away. The company works with a freight forwarder to arrange transport from the factory for the shipment. A trucking company, the freight forwarder, arrives at the factory, loads the order, along with orders from many other retailers, into a container. The container will not be opened until it arrives at its final destination, unless customs officials decide to open it and inspect it. The freight forwarder trucks the container to a port. The freight company has contracted with a liner shipping company, which must submit documentation, socalled “manifest data”, about the shipment to government authorities in the exporting and importing countries. The container is loaded onto a ship and shipped to its destination port, or a port where it is transhipped and picked up by another liner ship. When the vessel arrives at its destination port, large cranes are used to unload the containers of cargo. Workers load the containers onto a truck. If the destination is far away the containers are often transported by trains. The truck arrives at its destination.

2.2 The world fleet

World Shipping Council reports that there are approximately 4,900 container ships in today's global fleet and that there are approximately 600 new vessels on order. In an average year a large container ship travels about 290 000 km. To put this in perspective - this distance corresponds to more than seven times around equator. The average age of a container ship in the world fleet is 11.34 years (Asariotis et al., 2014).

Figure 2.5 shows the composition of the total world fleet. Container ships make up 12,8% of the world fleet in year 2014, which is a huge increase compared to 1.5% in 1980.



	1980	1990	2000	2010	2014
🟩 Other	4.5	7.5	9.4	7.2	11.2
⬜ Container	1.6	3.9	8.0	13.3	12.8
🟥 General cargo	17.0	15.6	12.7	8.5	4.6
🟨 Dry bulk	27.2	35.6	34.6	35.8	42.9
🟦 Oil tanker	49.7	37.4	35.4	35.3	28.5

Figure 2.5: World fleet by principal vessel types, 1980–2014 (Beginning-of-year figures, percentage share of dwt) (Asariotis et al., 2014).

Container liners

Container ships are distinguished into seven major size categories (see Table 2.1): small feeder, feeder, feedermax, panamax, post-panamax, new panamax and ultra-large (Vesterager, 2012).

Name	**Capacity (TEU)**
Ultra Large Container Vessel (ULCV)	14,501 and higher
New panamax	10,000–14,500
Post panamax	5,101–10,000
Panamax	3,001 – 5,100
Feedermax	2,001 – 3,000
Feeder	1,001 – 2,000
Small feeder	Up to 1,000

Table 2.1: Container ship size categories, based on capacity (TEU) (World Shipping Council, 2014a).

Container ships carry their containers both on and below deck. Deck-stowed containers on an 11,000 TEU vessel are stacked up to seven to eight high and nineteen across, inter-locked with fittings and secured by special lashings. Ships generally carry a mix of 20-foot and 40-foot boxes. Most can also carry 45-foot and 48-foot containers on deck and some can carry 53-foot containers as well.

Container ships have grown in size from just 1,500 TEU in 1976 to ships able to carry in excess of 12,000 TEU today, with some ships on order capable of carrying 18,000 TEU. Today's ships are not only able to carry more goods in one voyage, they are much more fuel efficient (World Shipping Council, 2014a).

Operators

From Table 2.2, we see that as per 1 January 2014, the largest container-ship operator in terms of container carrying capacity in TEU is MSC, based in Switzerland. It is followed by Danish Maersk Line and French CMA-CGM. Many of the ships deployed by the operators are in fact not owned by them, but leased from so-called "charter owners". In early 2014, it is estimated that about 60% of the order book of new container ships is on account of these charter owners, while the remaining 40% are ordered by the liner operators themselves; historically, the relationship used to be more in the range of 50:50 between operators and charter owners (Asariotis et al., 2014).

Rank	Operator	Country	Vessels	TEU
1	Mediterranean Shipping Company S.A.	Switzerland	461	2 609 181
2	Maersk Line	Denmark	456	2 505 935
3	CMA CGM S.A.	France	348	1 508 007
4	Evergreen Line	China	229	1 102 245
5	COSCO Container Lines Limited	China	163	879 696
6	Hapag-Lloyd Aktiengesellschaft	Germany	159	762 613
7	China Shipping Container Lines Comp. Ltd	China	134	750 644
8	Hanjin Shipping Company Limited	Korea	115	671 210
9	APL Limited	Singapore	121	629 479
10	United Arab Shipping Company (S.A.G.)	Arab states[2]	73	610 294

Table 2.2: The 10 leading liner companies, 1 January 2014 (Number of ships and total shipboard capacity deployed, in TEUs, ranked by TEU) (Asariotis et al., 2014).

2.3 Global market

According to the World Shipping Council liner shipping could lay claim to be the world's first truly global industry. Likewise it could claim to be the industry which, more than any other makes it possible for a global economy to work. It connects countries, markets, businesses and people, allowing them to buy and sell goods on a scale not previously possible.

The needs of a rapidly growing world population can only be met by transporting goods and resources between countries. The liner shipping industry has made this process more efficient and changed the shape of the world economy. This benefits consumers by creating choice, boosting economies and creating employment. Costs for the consumer are kept down and efficiencies are improved and this in turn minimizes impact on the environment as well World Shipping Council (2014a).

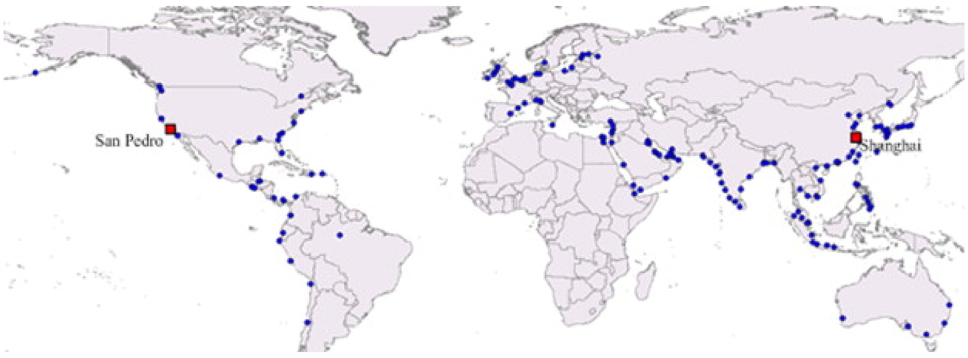


Figure 2.6: A global liner shipping network with 166 ports (Wang et al., 2013).

Global containerized trade grew by 4.6 % in 2013 taking total volumes to 160 million TEUs, up from 153 million TEUs in 2012. Together, intraregional (led by intra-Asian trade) and South–South trades accounted for 39.8 % of global containerized trade shipments in 2013, followed in descending order by North–South trade, the trans-Pacific, Far East–Europe, secondary East–West and transatlantic (Asariotis et al., 2014).

Rank	**Exporter**	**TEUs (Millions)**	**Importer**	**TEUs (Millions**
1	China	31.3	United States	17.6
2	United States	11.2	China	12.0
3	Japan	5.7	Japan	6.1
4	South Korea	5.2	South Korea	4.5
5	Taiwan	3.4	Germany	2.8

Table 2.3: Top 5 exporters and importers of containerized cargo in 2010 (World Shipping Council, 2014a).

Table 2.3 shows the top five exporters and importers of containerized cargo, respectively. On the export side, liner trade is dominated by countries in East Asia, with China in a class on its own exporting more than the four following countries together.

Trade route	**TEUs shipped (Millions)**
Asia - North America	22.0
Asia - North Europe	13.4
Asia - Mediterranean	6.2
North Europe - North America	4.6
Asia - Middle East	4.1

Table 2.4: Top 5 trade routes (TEU Shipped) 2012 (World Shipping Council, 2014b).

According to Asariotis et al. (2014) liner exports are highly concentrated, with the top ten exporting nations accounting for more than two-thirds of the total liner export value, and Greater China [3] account for 28% of the value of liner exports and 30% of the global volume of containerized exports. This is also confirmed by Table 2.4, where we can see that Asia is a part of four of the top five largest trade routes.

Rank	**Port**	**Country**	**TEUs (Millions)**
1	Shanghai	China	32.5
2	Singapore	Singapore	31.6
3	Hong Kong	China	23.1
4	Schenzen	China	23.0
5	Busan	South Korea	17.0

Table 2.5: Top 5 ports with most container traffic (TEU Shipped) in 2012 (American Association of Port Authorities, 2013).

Since ports handle a variety of traffic other than containerized shipments, one must rank based on container traffic in order to get a measure for the most active container ports, see Figure 2.5.

[3] Mainland China, Hong Kong S.A.R. and Taiwan, China.

The Top 50 container ports represent 30 countries demonstrating the truly global nature of the liner shipping business and the importance of the network of ports that facilitate timely and efficient ship and cargo movement (World Shipping Council, 2014a).

2.4 Benefits of liner shipping

International liner shipping is a sophisticated network of regularly scheduled services that transports goods from anywhere in the world to anywhere in the world at low cost and with greater energy efficiency than any other form of international transportation. In this section we highlight some of the benefits of liner shipping.

With rates for sea cargo transportation at approximately one-tenth of air freight rates, fewer accidents and less pollution, maritime transportation is regarded as a cheap, safe, and clean transportation mode, compared with other modes of transportation.

Efficiency. Liner shipping is the most efficient mode of transport for goods. In one year, a single large containership might carry over 200,000 container loads of cargo. While individual ships vary in size and carrying capacity, many container ships can transport up to 8,000 containers of goods and products on a single voyage. Similarly, on a single voyage, some car carrier ships can handle 7,600 cars. It would require hundreds of freight aircraft, many miles of rail cars, and fleets of trucks to carry the goods that can fit on one large liner ship (World Shipping Council, 2014a).

Global economic engine. Liner shipping connects countries, markets, businesses and people, allowing them to buy and sell goods on a scale not previously possible. Today, the liner shipping industry transports goods representing approximately one-third of the total value of global trade. Additionally, as a major global enterprise in its own right, the international shipping industry is responsible for millions of existing jobs and plays a crucial role in stimulating new jobs. It contributes hundreds of billions of dollars to the global economy annually thereby increasing gross domestic product in countries throughout the world. Moreover, as the lifeblood of global economic vitality, ocean shipping contributes significantly to international stability and security

Low environmental impact. Ocean shipping is the most carbon-efficient mode of transportation and produces fewer grams of exhaust gas emissions for each ton of cargo transported than air, rail, or road transport. In addition, new International Maritime Organization regulations establish strict standards for vessels' NOx, SOx, and particulate matter emissions. Also, the millions of containers that are used around the world are 98 % recyclable. Additionally, the CO2 emissions per ton cargo transported using maritime transport is significantly lower than road and rail transport (Karsten et al., 2015).

Chapter 3

Literature review

This chapter provides an overview of literature relevant to the Liner Shipping Network Design Problem (LS-NDP). As mentioned in Chapter 1, the volume of publications about service route planning is fairly limited, because of the confidentiality that often shrouds highly commercial information such as service patterns, alliance operations and marketing strategies. Meng et al. (2014) note the gap between existing academic studies and industrial practices within the industry. Kjeldsen (2011) points out that literature related to liner shipping is largely concerned with specific applications which is why the problem formulations and solution methods are much diversified.

As seen in Figure 1.3 it is common to distinguish between the problem of designing the routes, *the Network Design Problem* (NDP), and the deployment of the fleet, *the Fleet Deployment Problem* (FDP). The former is the problem of designing the routes, while the latter is concerned with assigning ships to the routes. In this paper the main focus is on designing the routes, while the fleet deployment problem will be solved implicitly as a part of this problem. Moreove, Meng et al. (2014) emphasize that the NDP is associated with the tactical-level problems and thus cannot be investigated in isolation. Because of this we have examined papers about both the NDP and the FDP, and some other decisions associated with liner shipping.

Since the LS-NDP consists of two fields of study namely *Liner Shipping* and *Network Design Problems*, we found it natural to divide the investigation of literature into two parts. In Section 3.1 we discuss how important aspects of liner shipping are handled in the literature, while we in Section 3.2 review literature about general network design problems. Throughout the review we point out how previous research about the LS-NDP relates to the work in this report.

3.1 Aspects of liner shipping

In this section we will go through key features one must adhere to when constructing routes in liner shipping, and see how these are dealt with and discussed in the literature.

Sailing speed and fuel consumption. The sailing speeds have a significant impact on the total operating costs. According to Meng et al. (2014) many researchers have used the power of three relation in their studies, i.e. that the daily bunker consumption is approximately proportional to the speed cubed.

Socalled *slow steaming*, i.e. the practice of operating transoceanic container ships at significantly less than their maximum speed, is an increasingly popular method used to save money on fuel. Slow steaming is currently often seen as a win-win-win proposition – delivering value through a balanced "people-profit-planet" approach, but it is unclear if under a changing economic and business environment slow steaming will sustain (CNSS, 2013).

According to Meng et al. (2014) most studies of LS-NDP and FDP have assumed that ships sail at predetermined speed, which implies that they do not take into account the possibility of slow steaming. For instance Fagerholt (1999), Christiansen and Nygreen (1998) and Agarwal and Ergun (2008) relied on the assumption that all ships have the same sailing speed. Alvarez (2009) discretized the speed range and considered each combination of ship type and speed interval a separate ship type. This was done because a containership may vary its speed within a certain range.

In this study we assume fixed sailing speed for each ship type, and thus the possibility of slow steaming is not taken into account.

Frequency requirements. Christiansen and Nygreen (1998) write that recent efforts in liner container shipping have generally adopted the weekly frequency because global liner shipping companies usually provide weekly shipping services. Examples are Fagerholt (1999), Christiansen and Nygreen (1998) and Agarwal and Ergun (2008) who have a weekly frequency requirement on the routes. In Reinhardt and Pisinger (2011) a port is allowed to be visited less than once a week. However, they do not assume weekly demand as we do in this thesis. In this report we assume weekly frequency requirements.

Repositioning of empty containers. A serious challenge associated with containerized shipping is the repositioning of empty containers. Because of the huge imbalance in trade volume on some routes, carriers need to reposition empty containers, incurring significant costs. According to ROI (2002), a 10% reduction in equipment and repositioning costs can potentially increase profitability by 30-50% (Agarwal and Ergun, 2008). The model provided by Agarwal and Ergun (2008) has, with some modifications, the flexibility to incorporate empty container repositioning. Cheung and Chen (1998) study empty container repositioning, however, the paper only considers the movement of empty containers on a given network. In this report we do not consider empty container repositioning.

Alliance strategies. In their latest report United Nations Conference on Trade and Development (UNCTAD) state that building shipping alliances is becoming an important strategy for shipowners to control costs and maximize capacity utilization on larger ships (Asariotis et al., 2014). Agarwal and Ergun (2010) present three factors that drive carriers to adopt solutions outside of their traditional business practices and collaborate with their competitors: (i) liner shipping is a capital-intensive industry; (ii) large containerships produce economies of scale, however, they require a longer period for container accumulation, resulting is a less frequent service; (iii) alliances help carriers to explore new markets and enhance their service scope. Song and Panayides (2002) applied cooperative game theory to analyze cooperation among members of liner shipping strategic alliances, producing a conceptual framework that enhanced understanding of interorganizational relationships and decision-making behavior in the liner shipping sector.

The possibility of collaborations and alliances is not taken into account in this report. We assume one single actor that operates individually with no interference with other operators.

Fleet composition. Some papers assume a *homogeneous fleet*, while some arrange for ships that differ in terms of capacity, speed and cost structure, i.e. a *heterogeneous fleet*. An example of the former is Sambracos et al. (2004) who carried out a case study on the feeder ship route design used to dispatch small containers in the Aegean Sea, from one depot to 12 other ports. They assumed a homogeneous fleet that aimed to meet the minimum operating costs, including fuel consumption and port changes. Agarwal and Ergun (2008) on the other side incorporated a heterogeneous fleet in their model.

Perakis and Jaramillo (1991) and Jaramillo and Perakis (1991) implicitly and unrealistically assumed that the number of ships allocated on a route was a continous rather than a integer decision variable. Powell and Perakis (1997) presented an IP model to remedy this unrealistic assumption.

In the model presented in this report we also model the number of ships on a route an integer variable, and we allow a heterogeneous fleet with different ship types.

Route structures. A huge challenge in liner shipping network design is how to allow flexible route structures. One of the main reasons for this is that complex route structures complicate the calculation of the transhipment cost. Among existing literature the models either allow only simple routes or a mix of butterfly routes and simples routes.

Agarwal and Ergun (2008) model butterfly routes by using a time-space graph where a port can be visited several times as long as the visit is not on the same weekday. Butterfly routes are routes where at least one port is visited more than once. Another way to put it is a cargo route that encompass a combination of cycles rather than a single cycle. However, they do not consider the transhipment costs that can occur in the intermediate port. Reinhardt and Pisinger (2011) present a mathematical formulation which allows butterfly routes and also takes into account the transhipment cost. Transhipment is further discussed in the next paragraph. Note that Reinhardt and Pisinger (2011)'s definition of butterfly routes is more restrictive than the definition used in this report - they define butterfly routes as routes where one port is visited twice.

We have not found any articles that provide models that enable route structures with more than two visits to the same ports, nor to have several ports in a route that are visited more than once. In this report we present two different approaches that enable more flexible route structures; the *Flower model* and the *Chain approach.* The former allows for multiple visits to one single port in a route, while the latter allows a route to have several ports that can be visited twice. The models presented in this report are therefore more general than the one presented by Reinhardt and Pisinger (2011) in terms of route structures.

Transhipment and transhipment costs. Transhipment is the shipment of goods to an intermediate destination, then to yet another destination. Reasons for transhipment can be for *consolidation* or *deconsolidation* purposes, i.e. to combine small shipments into a large shipment or divide the large shipment at the other end, respectively. Transhipment usually takes place in transport hubs.

A few decades ago the use of transhipment was much less common, and older articles such as Rana and Vickson (1991) do not include transhipment in their route planning. The use and influence of transhipment on the liner shipping network design is described by Notteboom and Rodrigue (2008). Whether or not transhipment operations are allowed between the routes differ from paper to paper. In a classification done by Meng et al. (2014) 8 out of 12 of the papers on fleet deployment did not consider transhipment.

In Agarwal and Ergun (2008) transhipment is considered, however they do not include the associated cost. They explain this with the tremendous increase it will cause to the size of the graph. In their concluding remark they write: "Our results indicate high percentage utilization of ships' capacities and a significant number of transhipment in the final solution". When the transhipment cost is not taken into account the transhipment in the optimal solution will probably be inexpendiently big.

Reinhardt and Pisinger (2011) argue that transhipment of goods is frequently occurring in liner shipping and the associated cost should not be ignored when designing the network. They claim that their paper presents the first exact solution methods to the LS-NDP with transhipment cost.

More complex route structures allow for transhipment within the same route, because a route can tranship to itself because it visits the same port multiple times. This type of transhipment is denoted *internal transhipment* in this report. The cost of internal transhipment is also included in Reinhardt and Pisinger (2011). They write about internal transhipment; "In models where each port is represented by one vertex at the points where two cycles are connected a transhipment from an early visit of a vessel to a later visit of the same vessel can occur. [..] this results in complications in the calculation of transhipment costs." The internal transhipment offers challenges in the modeling process because it is quite comprehensive to be able to capture the internal transhipment happening in a port.

To capture the internal transhipment Reinhardt and Pisinger (2011) enumerate the edges on the route and marks the first and last edge on the entire route. This is done to be able to distinguish between two visits to the same centerports. [1]

Transhipment and transhipment costs, for both external and internal transhipment, are incorporated in our models. Capturing the value of the internal transhipment is very complicated in the Flower model and how this is done is thoroughly described in Chapter 6.

Uncertain future demand. Meng et al. (2014) classify a number of papers as to if they incorporate deterministic or stochastic demand in the model. Of the in total 14 papers they reviewed 3 of them model stochastic demand while 11 assume deterministic demand. Meng and Wang (2011) analyzed a multi-period fleet planning problem. They assumed that container shipment demand varied between periods, and that the liner shipping company could sell, purchase, charter out ships at the beginning of each period.

Meng et al. (2012) modeled a two-stage stochastic integer programming model in order to take into account their assumption of uncertain demand as a random variable with known probability distribution.

[1]In Reinhardt and Pisinger (2011) the term centerport corresponds to what we in this report call the *depot*, i.e. a port visited more than once in a butterfly route, or the arbitrarily chosen reference port in a simple route.

In this paper we assume a deterministic weekly demand.

Pollution. Some studies have focused on the CO2 emissions of ships. Psaraftis and Kontovas (2010) analyzed the implications of various maritime emission-reduction policies for maritime logistics. They concluded that important trade-offs may have to be made between the environmental benefits between associated with such measure as a reduction in steaming speed and changes in the number of vessels in fleet, and more conventional logistics attributes such as in-transit inventory holdings.

In this study we have not considered pollution explicitly. The considerations we take regarding this is limited to the fact that an optimal economical solution will promote short sailing distances and minimal fleet usage - and is in this way incentive compatible with an objective to reduce the pollution.

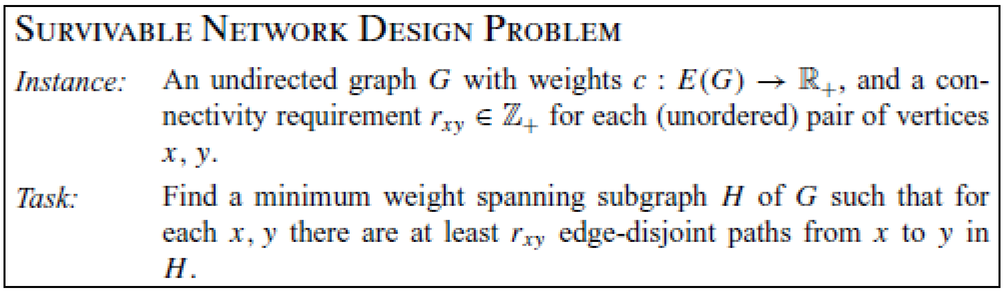
Transit time. The transit time is the time it takes a commodity to travel from origin to destination. Transit time is counted in days and allowed transit times may vary from one day to several months (Brouer et al., 2014). Offering short transit time is a major competitive factor for liner shipping companies, especially when the goods transported are time sensitive, such as fashion and computers (Notteboom, 2006). Implications of travel time restrictions are not well-studied in connection with LSND, but have recently been studied in connection with related problems (Karsten et al., 2015). According to Notteboom (2006) it is very difficult for link-based formulations to incorporate service-level considerations in terms of the O–D transit time. Brouer et al. (2014) states that in liner shipping there is an inherent trade-off between reducing the bunker consumption through speed reduction and achieving competitive transit speed for cargo.

Karsten et al. (2015) studies the consequence of neglecting the transit time restrictions in existing networks, and shows that it is essential to include transit time constraints in LSND.

In this paper we extend the models to incorporate a maximum transit time for each order in the network.

3.2 Network design problems

Cieslik (2009) defines a *network* as a connected graph with a length-function. The general *Network Design Problem* (NDP) is a combinatorial problem where the objective is to find the optimal subgraphs that satisfy certain connectivity requirements. In Figure 3.1 a formal definition of the general NDP is presented:



Survivable Network Design Problem

Instance: An undirected graph G with weights $c : E(G) \to \mathbb{R}_+$, and a connectivity requirement $r_{xy} \in \mathbb{Z}_+$ for each (unordered) pair of vertices x, y.

Task: Find a minimum weight spanning subgraph H of G such that for each x, y there are at least r_{xy} edge-disjoint paths from x to y in H.

Figure 3.1: Formal definition of the general NDP (Korte and Vygen, 2006).

The multi-commodity network flow problem is an important sub-problem in several heuristics and exact methods for designing route networks for container ships (Karsten et al., 2015).

Complexity. Network design problems are of nontrivial complexity. In fact, Johnson et al. (1978) prove that the NDP is NP-complete even in the simple case where all edge weights are equal and the budget restricts the choice to spanning trees. The LS-NDP is NP-hard (Agarwal and Ergun, 2008). This means that we cannot expect to find a polynomial-time algorithm that will produce the optimal solution.

Applications. Network design problems span a wide range of applications, from modeling the evolution of species in biology to modeling soap films for grids of wires; from the design of collections of data to the design of heating or airconditioning systems in buildings; and from the creation of oil and gas pipelines to the creation of communication networks, road and railway lines (Cieslik, 2009).

Schmid (2014) presents a model formulation for the bus rapid transit route design problem, given a fixed number of routes to be offered. They use a decomposition strategy, where route design and the determination of frequencies and passenger flows are dealt with separately. Due to the complexity of the problem they propose a hybrid metaheuristic based on a combination of Large Neighborhood Search (LNS) and Linear Programming (LP), and the decision upon the design of routes is handled using LNS. The proposed algorithm is able to obtain high quality solutions within short computational times.

Many of the problems associated with liner shipping are similar to those addressed by airlines: among them aircraft routing and airline fleet assignment. Barnhart et al. (1998) present a single model and solution approach to solve simultaneously the fleet assignment and aircraft routing problems. Their approach is robust in that it can capture costs associated with aircraft connections and complicating constraints such as maintenance requirements. However, according to Meng et al. (2014), the problems associated with liner container shipping are generally more challenging than the corresponding problems for airlines, because of its more complex operations. For instance, a flight generally has only two one or two legs and a liner service route may have 10-20 legs.

LS-NDP. Maritime transportation has attracted much less research effort than other transportation areas, such as public transit analysis, airline management, and general vehicle routing problems (Psaraftis, 1999). Among the research done on maritime transportation, even fewer studies are devoted to liner shipping, compared to industrial and tramp shipping.

Heaver and Uyeno (1987) proposed an analytical model for planning liner container shipping operations. They first enumerated all possible ship routes and then developed a set-partitioning model to choose the optimal set of ship routes that would satisfy the shipping requirements.Fagerholt (1999) contributed a pioneering study in the feeder network category. He proposed a set-partitioning model by enumerating all possible shipping service routes and combining these single shipping service routes if possible. The model relied on the assumption that all ships have the same sailing speed, while Fagerholt (2004) extended the model to address a heterogeneous ship fleet with a given cost structure, capacity and speed for each type.

Recall that the LS-NDP is NP-hard. In Reinhardt and Pisinger (2011) they developed an exact B&C algorithm to solve instances up to 15 ports. As far as we are concerned this appears to be the state of the art when it comes to exactly solving LS-NDP instances.

The models in this paper are arc-flow formulations that construct the routes they solve the problem, i.e. not a path-flow formulation like the one presented by Heaver and Uyeno (1987). Another characteristic of the models are that they are not tailored towards a special type of network, and are thus more general than the one Fagerholt (1999) presents.

Chapter 4

Problem description

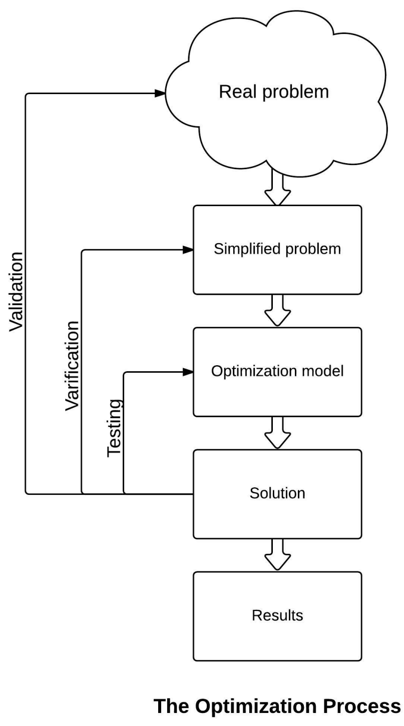


Figure 4.1: The stages in the optimization process. The problem description captures the essence of the real problem into a simplified problem.

The previous chapters have given an introduction to the purpose of this report, the LS-NDP and the liner shipping industry. We have described the challenges related to the problem - both in terms of reality representation, uncertainty and complexity. In this chapter we boil this down to a problem description with accompanying assumptions and simplifications. The *problem description's* role in the optimization process is illustrated in Figure 4.1.

We study the Liner shipping network design problem (LS-NDP) which consists of designing optimal vessel routes with an associated fleet of vessels, so that the forecasted demand is satisfied with a minimal cost for the company. The problem is considered from the point of view of a ship owner operating a heterogeneous fleet of vessels.

In order to reflect a realistic situation we take into account multiple pickups and deliveries, the cost of transhipment and route dependent capacities. In Chapter 5 we present a basic model of the problem - only allowing simple routes. In Chapter 6 and Chapter 7 we extend this model, the Basic model, and present two models that incorporate more complex route structures, namely the Flower model and the Chain model. In Chapter 8 we present extensions of all these three models where we introduce transit time. When nothing else is specified the assumptions and simplifications made apply on a general basis, i.e. for all six models.

4.1 Modeling assumptions

Minimization of costs. The goal of the ship owner is to minimize costs by designing the optimal routes. We consider three types of costs in this model:

- Sailing costs
- Transhipment costs
- Visiting costs

There are, obviously, several other types of costs that accrue for a liner shipping company. However, we have taken into account the costs based on their magnitude and also the impact they will have on differentiating different route structures from each other. We find this is a reasonable assumption considering that bunker costs constitute up to 60% of the total ship operating costs. Other costs we could have included are: the cost of using a vessel, maintenance costs, route costs, steaming costs, and the cost of empty container repositioning.

Deterministic demand. We assume a known deterministic constant weekly demand. This means that the model must ensure that if there is a demand D_{od} from port o to port d, D_{od} must be transported from o to d every week.

Weekly service frequency. A weekly service frequency is a convention in the industry. It should be noted that there are exceptions, e.g., Maersk Line provides a daily Asia-Europe shipping service called Daily Maersk, which is more competitive than weekly services. In our model we make the simplification of a weekly frequency requirement on all routes.

Constant sailing speed. We assume the speed of each ship type to be constant. This clearly is a limiting simplification when we know that in reality the sailing speeds of ships have a significant impact on the total operating costs. An increase of just a couple of knots results in a dramatic increase in bunker consumption (Notteboom and Rodrigue, 2008). This means that our model does not facilitate the possibility of slow steaming. Slow steaming is a cost-effective option to reduce CO2 emissions in the short-term (Cariou, 2014).

Only one ship type per route. Because of the constant weekly demand there will have to be only one ship type per route to comply with the weekly frequency demand. This is in accordance with the tendency towards ships remaining on the same route throughout the planning horizon (Kjeldsen, 2011).

Steady state model. Since all demands must be picked up and the weekly demand is constant, the frequency of a route is also constant. This implies that it is not necessary to specify a time period in the model, nor the origin and destination ports for the vessels, because the behavior of the system will be equal at all times.

Heterogenous fleet. Our model takes into account a heterogeneous fleet. The ship types can differ by the following characteristics:

- Speed
- Capacity
- Sailing cost
- Visiting costs at the ports

Symmetric sailing distances and triangle inequality. The network is expected to be undirected, i.e. we presume symmetric sailing distances. We assume that the direct connection between two ports will always be the fastest route between them.

Transhipment. Transhipment and transhipment cost are included in the model, however the transhipment cost is independent of how long time the cargo spend at a port. The transhipment of containers at a container port or terminal is defined as the number of containers (expressed in TEU) of the total container flow that is handled at the port and, after temporary storage in the stack, transferred to another ship to reach their destinations. There can be transhipment in all ports, and we assume heterogeneous types.

Calculation of sailing costs. Because of the weekly frequency requirement the number of vessels assigned to a route has to be greater than the number of weeks it takes to sail the route for that particular ship type. E.g. if a route takes 10.8 weeks to sail it needs 11 vessels to fulfill the frequency requirement. Since we assume constant sailing speed, i.e. if a ship is assigned to a route it will sail in the predefined speed at all times, this implies that if a route takes 10.8 weeks to sail and needs 11 ships - the sailing cost will be the same as if the route

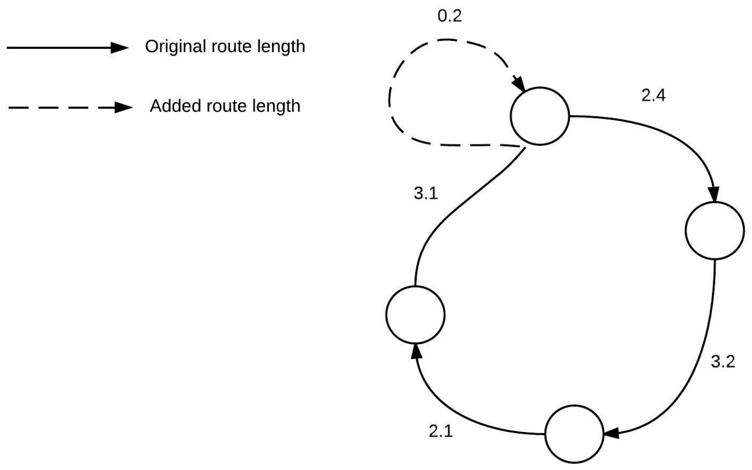


Figure 4.2: Calculation of sailing costs. All vessels sail at constant speeds. This means that the sailing cost is related to how many vessels the route needs, and not directly to the length of the route.

would have taken 11 weeks to sail, because all the 11 vessels have to sail "all the time" anyway. This corresponds to that the vessels have to take an extra loop to ensure that the route takes an integer number of weeks to sail. This is illustrated in Figure 4.2. Since all vessels sail at constant speed the calculation of sailing cost is related to how many vessels the route occupies, and not directly to the length of the route.

Route structures. As has been stated before; in this report six models for the LS-NDP are presented; the Basic model, the Flower model and the Chain model, along with the transit time extensions of these. So far all the assumptions mentioned above apply for all of them. Except from the transit time extension, the manner in which the models differ is what kinds of route structures they enable. Exactly what kinds of route structures the different models allow are thoroughly explained in the models' respective chapters. Figure 4.4 show different route structures along with their names and characterizations.

4.2 Problem in context

Kjeldsen (2011) has developed a classification scheme for the LS-NDP. The classification scheme has eighteen characteristics with anywhere from two to eight possible options (Kjeldsen, 2011). Twenty-four articles within ship routing and scheduling published between 1969-2010 are classified. The purpose of the classification is to enable a more informed choice of which model to use in a specific case. Table 4.2 shows the classification scheme and the 24 classified articles given in Table 4.1. To put our models into context we have marked the classification options that apply for them with an *, so to see where the models are placed in the territory of LS-NDP.

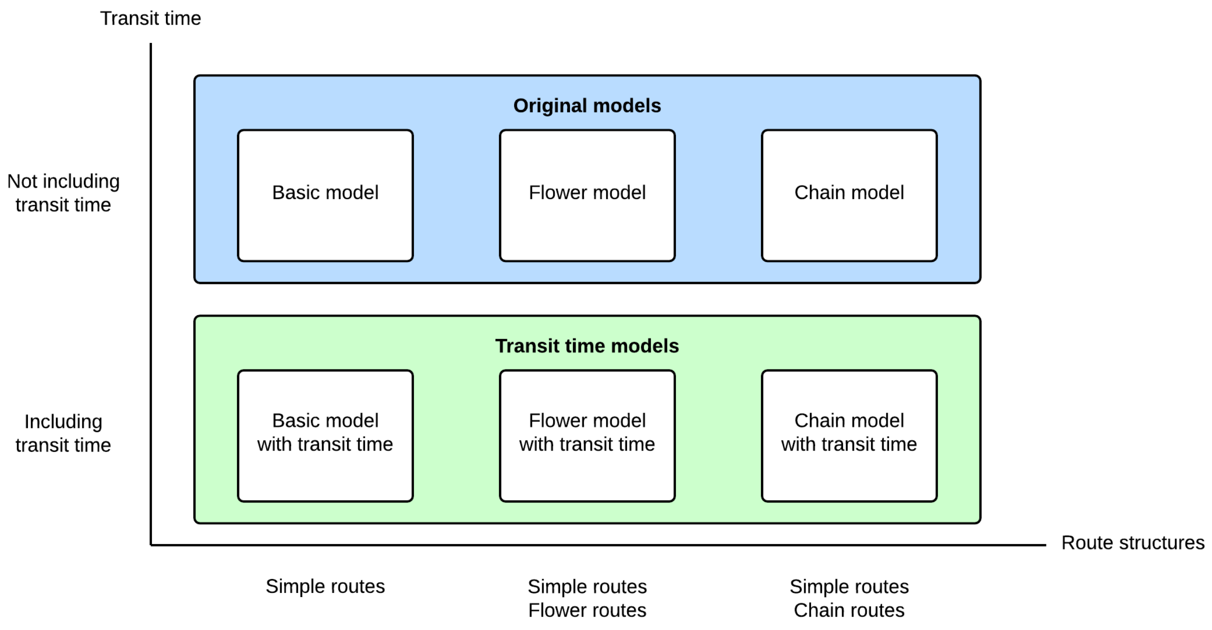
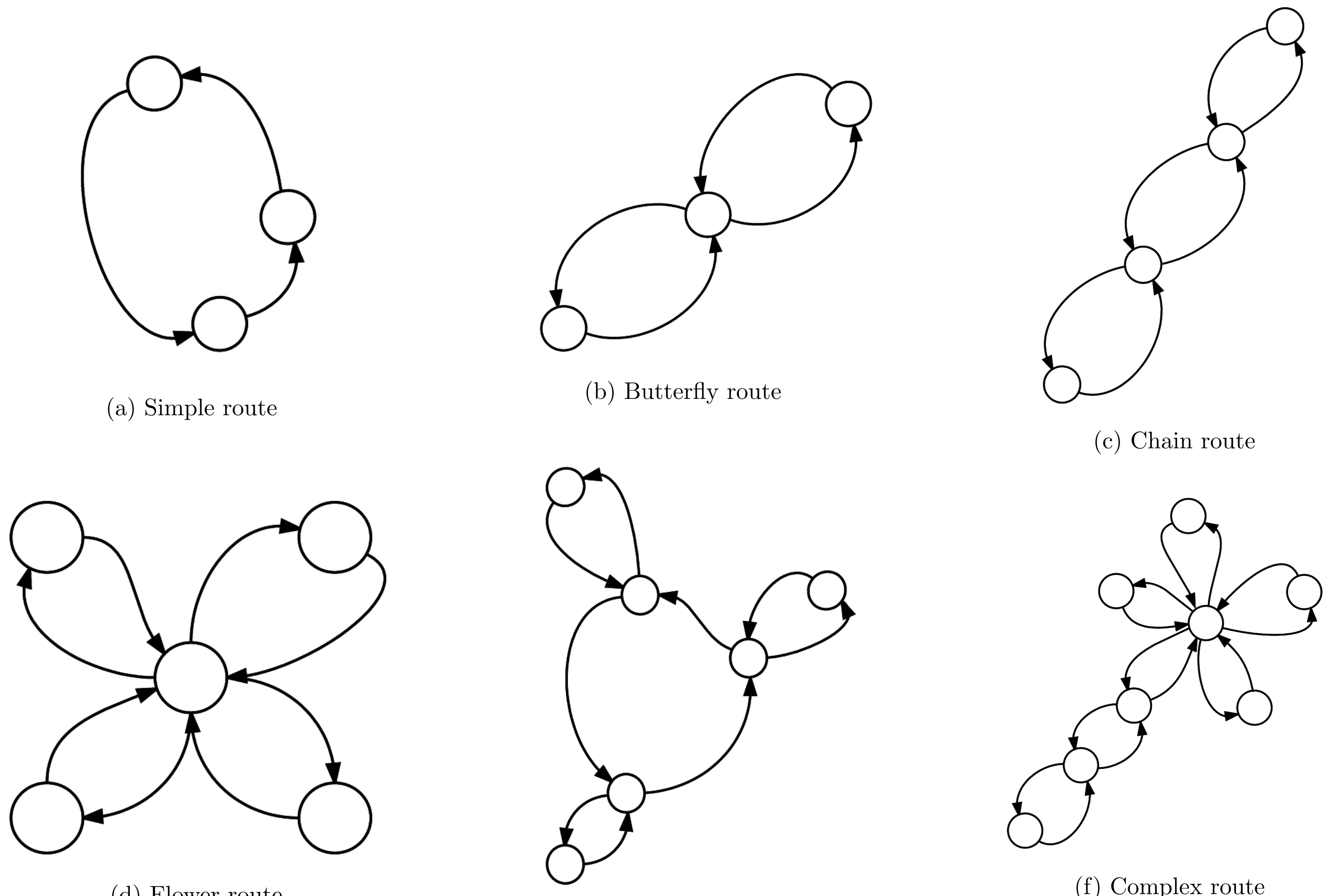

Transit time
Not including transit time
Including transit time
Original models
Basic model
Flower model
Chain model
Transit time models
Basic model with transit time
Flower model with transit time
Chain model with transit time
Route structures
Simple routes
Simple routes Flower routes
Simple routes Chain routes

Figure 4.3: The functionality of the six models presented in this paper differs in two dimensions; whether or not they incorporate transit time and what kind of route structures they enable. The Flower and Chain models enable different kinds of complex routes.

Even though it has been written several important articles since this classification was performed (e.g. Reinhardt and Pisinger (2011)), the classification says something about what has traditionally been emphasized and what has been neglected in liner shipping routing and scheduling models. An observation from the categorization is that only 3 of the 24 classified articles take into account transhipment cost. Based on the importance of transhipment described in the former chapter, this is an indication that there is a need for more models incorporating this.

1	Kydland (1969)	13	Fagerholt (2004)
2	Olson et al. (1969)	14	Sambracos et al. (2004)
3	Boffey et al. (1979)	15	Reinhardt et al. (2007)
4	Lane et al. (1987)	16	Shintani et al. (2007)
5	Rana and Vickson (1991)	17	Sigurd et al. (2007)
6	Pesenti (1995)	18	Agarwal and Ergun (2008)
7	Cho and Perakis (1996)	19	Alvarez (2009)
8	Fagerholt (1999)	20	Fagerholt et al. (2009)
9	Fagerholt and Lindstad (2000)	21	Imai et al. (2009)
10	Bendall and Stent (2001)	22	Karlaftis et al. (2009)
11	Moura et al. (2002)	23	Yan et al. (2009)
12	Ting and Tzeng (2003)	24	Chuang et al. (2010)

Table 4.1: Numbers assigned to articles in Kjeldsen (2011)



(a) Simple route

(b) Butterfly route

(c) Chain route

(d) Flower route

(e) Complex route

(f) Complex route

Figure 4.4: Different route structures: (a): A *Simple route* is a route where no ports are visited more than twice in the same route. (b): A *Butterfly route* is a route where *one and only one* port in the route is visited *twice*. (c): A *Chain route* is a route where *at least one* port is visited twice, and no ports are visited more than twice. (d): A *Flower route* is a route where *one and only one* port is visited more than once. (e) and (f): *Complex routes* are routes where at least one port is visited at least twice. Note that a butterfly route is also a flower and chain route, and flower and chain routes are also complex routes.

No.	Characteristics	Options	Articles
1	Number of starting points	**None**	18-20, 23
		One	1, 6, 8-14, 16, 17, 22-24
		Multiple	3-5
2	Type of operation	Delivery	6, 9
		Pick-up	8, 13
		Pick-up and delivery separated	20, 24
		Pick-up and delivery intervowen	1-5, 7, 10-12, 14-19, 21-23
3	Nature of demand	**Deterministic**	2, 4-9, 11-23
		Stochastic	1, 24
		Dependent on service	3, 10
4	Scheduling constraints at the ports	Time of service fixed in advance	2
		Time windows	4, 9, 11, 12, 17, 18, 20, 22, 23
		No restrictions	1, 3, 5-8, 10, 13-16, 19, 21, 24
5	Number of ships	**Fixed**	3, 5, 12, 13, 15, 21, 23, 24
		Changeable - constant over scheduling period	1, 4, 6, 8-11, 14, 16-19, 22
		Changeable - changes over scheduling period	2, 7, 20
6	Fleet composition	**Heterogenous**	2, 4-9, 11-13, 15-20, 23
		Homogeneous	1, 3, 10, 14, 21, 22, 24
7	Cruising speed	Yes	12, 16, 19
		No	1-11, 13-15, 17, 18, 20-24
8	Demand splitting	**Allowed**	1, 2, 5, 9-11, 15, 18, 19, 23
		Not allowed	3, 4, 6-8, 12-14, 16, 17, 20-22, 24
9	Partial satisfaction of demand	Allowed	1, 3, 5, 6, 10, 16, 18, 19, 23, 24
		Not allowed	2, 7-9, 11-15, 17, 20-22
10	Number of capacity types	**One**	3-8, 10-14, 16, 17, 19-24
		Multiple	1, 2, 9, 15, 18
11	Cargo transhipment	**Allowed**	11, 14, 15, 18, 19, 23, 24
		Not allowed	1-10, 12, 13, 16, 17, 20-22
12	Number of routes per ship	**One**	1, 4-5, 12, 15, 17-19, 21-24
		Multiple	2, 6-11, 13, 16, 20
13	Planning horizon	Defined- ships must finish routes	3, 5, 8-18, 20, 22
		Defined - ships need not finish routes	2, 4, 7, 19, 21, 23
		Undefined	1, 6, 24
14	Ships required to be empty	Yes	1, 5, 8-10, 13, 14, 17, 20, 22, 24
		No	2-4, 6, 7, 11, 12, 15, 16, 18, 19, 21, 23
15	Port precedence requirement	Exists	1, 2, 5, 10, 17, 19, 20
		None	3, 4, 6-9, 11-16, 18-22
16	Ship-port compatibility	Exists	2, 7, 11, 17, 19, 20
		None	1, 3-6, 8-10, 12-16, 18, 21-24
17	Cost types	Fixed costs	
		-in operation	1, 6-8, 10, 11, 16, 17, 19, 21
		-in lay-up	2, 7, 19
		Variable costs	
		-steaming costs	1, 2, 4, 5, 7, 8, 10, 11, 13-20, 22-24
		-port entry charges	1, 4, 5, 8, 10, 11, 13, 14, 16, 18, 23
		-time spent in port	4, 18, 23-24
		-cargo operation	1, 2, 4, 10, 19, 20, 23
		-transhipment	15, 19, 23
		Cost of unserviced demand	19
18	Objective	**Minimize costs**	4, 7-9, 11, 13-15, 17, 19, 20, 22
		Maximize profits	1-3, 5-7,10, 16, 18, 23, 24
		Minimizing environmental impact	

Table 4.2: Schematic overview of Kjeldsen (2011)'s classification scheme and the 24 classified articles. Table 4.1 shows which articles the numbers correspond to. The options marked yellow are the options that apply for the models presented in this thesis.

Chapter 5

Basic model

In this chapter, one of our formulations of the LS-NDP, the Basic model, is presented. The Basic model only allows simple routes which means that a single route can not visit a port more than once. However, a port can be visited several times by different routes. A network of simple routes is illustrated in Figure 5.1. As the name indicates the Basic model serves as a *base model*, i.e. the other models in this thesis build upon the formulation of this model. Figure 4.3 in Section 4.1 shows how the Basic model relates to the other models that are presented in the subsequent chapters.

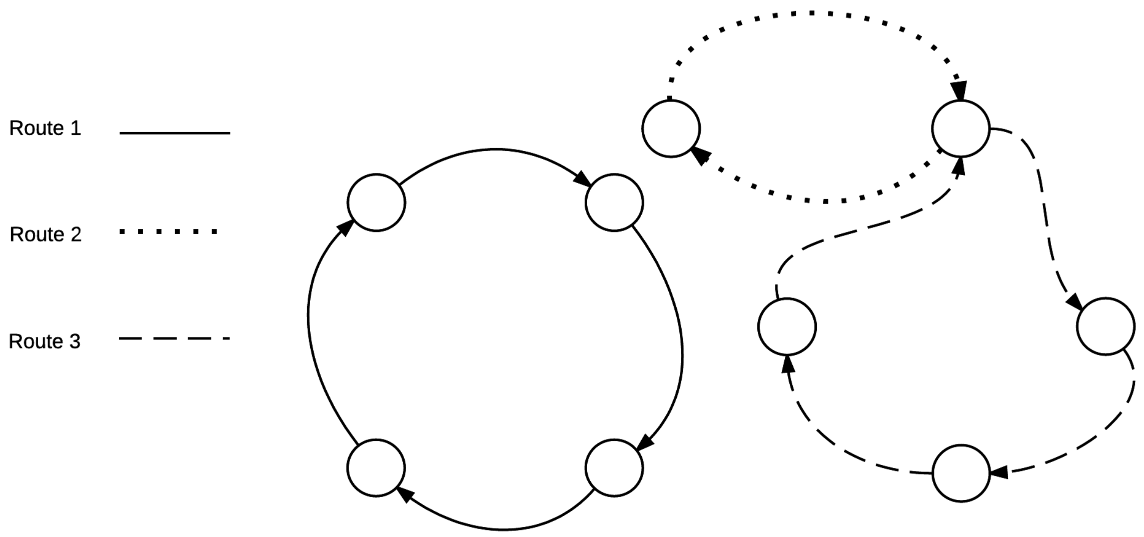


Figure 5.1: In a simple route-network a route does not visit any port more than once in the same sequence. However, a port can be visited by several routes. In the network depicted, transhipment can happen in the port visited by both Route 2 and Route 3.

This chapter is structured as follows: In Section 5.1 the mathematical formulation of the Basic model of the LS-NDP is presented. Because of their complexity and need for explanations, the subtour elimination constraints and linearizations are presented separately in Section 5.2 and Section 5.3, respectively.

5.1 Mathematical formulation of the Basic model

There is no standard mathematical formulation for the liner shipping problem since each liner service has specific constraints based on strategic decisions (Reinhardt and Pisinger, 2011). In this section the proposed mathematical formulation for the Basic model of the LS-NDP is presented, along with explanations of the constraints and modeling choices. For the compact formulation of the Basic model see Appendix A.2.

Notation. The notation is based on lower-case letters to represent indices and decision variables and capital letters to represent constants. Normally, the variables are only defined for some combinations of subscripts. To limit the level of detail in the mathematical formulation, we regard the none-defined variable combinations as eliminated from the model.

A note on route indexing. The model is an arc-flow formulation and thus it is going to *construct* the routes, i.e. it does not have a set of predefined routes to choose from. Even though the number nor the composition of the routes in the optimal solution are (of course) not known beforehand, the model still needs a way of referring to a specific route. This is resolved this by introducing the set of routes, R, stretching from 1 to the maximum number of routes, and a binary variable y_{kr} that is 1 if route r is sailed by ship type k. This set of routes creates a lot of symmetry in the problem, because which index a route gets is in reality totally irrelevant for the quality of the solution. Symmetry breaking constraints (SBCs) are discussed separately in Section 9.1, and in Chapter 12 their effect is evaluated to see which of them should be included in the models. What number is set to be the maximum number of routes also has a significant impact on the solution time. How this parameter should be set is discussed in Section 12.1.

Sets and indices

$\mathcal{P}$	- Ports, i,j	
$\mathcal{A}$	- Arcs, (i,j)	$i, j \in \mathcal{P}$
$\mathcal{D}$	- Demands, (o,d)	$o, d \in \mathcal{P}$
$\mathcal{K}$	- Ship types, k	
$\mathcal{R}$	- Routes, r	$r \in \{1.. \sum_{k \in \mathcal{K}} M_k\}, k \in \mathcal{K}$

Parameters

D_{od}	- Weekly demand from port o to port d	$(o,d) \in \mathcal{D}$
C_i^T	- Transhipment cost per unit port i	$i \in \mathcal{P}$
C_{ik}^P	- Visiting cost (per visit) at port i for ship type k	$i \in \mathcal{P}, k \in \mathcal{K}$
C_k^S	- Sailing cost per week for ship type k	$k \in \mathcal{K}$
V_k	- Velocity (NM/Weeks) of ship type k	$k \in \mathcal{K}$
T_{ij}	- Distance (in NM) on arc (i,j)	$(i,j) \in \mathcal{A}$
T_{ijk}	$= T_{ij}/V_k$. Sailing time on arc (i,j) for ship type k	$(i,j) \in \mathcal{A}, k \in \mathcal{K}$
S_k	- Capacity of ship type k	$k \in \mathcal{K}$
M_k	- Number of vessels of ship type k in fleet	$k \in \mathcal{K}$
N	- Number of ports	
L	- Max route length (NM)	
W	- Max route time (Weeks)	

Variables

x_{ijkr}	- 1 if a vessel of type k is used on arc (i,j) by route r, 0 otherwise
y_{kr}	- 1 if a vessel of type k is used on route r, 0 otherwise
f_{ijodkr}	- Flow of the demand (o,d) transported by a vessel of type k on arc (i,j) in route r
t_{iodr}	- The amount of demand (o,d) that is transhipped in port i in route r
a_{kr}	- The number of vessels of ship type k sailing route r
b_{ik}	- The number of times port i is visited by ship type k each week

Objective function

$$minimize \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} C_k^S \cdot\ a_{kr} + \sum_{i\in\mathcal{P}}\sum_{r\in\mathcal{R}} C_i^T \cdot t_{iodr} + \sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}} C_{ik}^P \cdot\ b_{ik} \tag{5.1}$$

The objective function (5.1) minimizes total cost. The cost function of the LS-NDP consists of the sailing costs of the routes, transhipment costs in hub ports and visiting costs in the ports.

Constraints

Route constraints

$$\sum_{k\in\mathcal{K}} y_{kr} \leq 1, \qquad r \in \mathcal{R} \tag{5.2}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} T_{ij} \cdot x_{ijkr} \leq L \cdot y_{kr}, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{5.3}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} T_{ijk} \cdot x_{ijkr} \leq W \cdot y_{kr}, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{5.4}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} x_{ijkr} - y_{kr} \geq 0, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{5.5}$$

Constraints (5.2) state that there can only be one ship type assigned to each route, in order to ensure the weekly demand frequency. Constraints (5.3)-(5.4) specify the maximum length (in NM) and total time (in weeks) a route can have, respectively. They also ensure that y_{kr} is set to 1 if ship type k sails route r, while constraints (5.5) force y_{kr} to 0 if the opposite is the case.

Flow constraints

$$\sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(f_{ijodkr} - f_{jiodkr}) = 0, \qquad j \in \mathcal{P}, (o,d) \in \mathcal{D}, j \neq o, j \neq d, \tag{5.6}$$

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} f_{ojodkr} = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{5.7}$$

$$\sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} f_{idodkr} = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{5.8}$$

$$\sum_{i\in\mathcal{P}} x_{ijkr} - \sum_{i\in\mathcal{P}} x_{jikr} = 0, \qquad j \in \mathcal{P}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5.9}$$

Constraints (5.6) are demand flow conservation constraints in the transition ports. Constraints (5.7)-(5.8) ensure that all cargoes are loaded and unloaded in their origin and destination ports, respectively. Constraints (5.9) are flow conservation constraints for the vessels and ensure that an equal number of ships enters and leaves a port.

Transhipment constraints

$$t_{iodr} \geq \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}(f_{ijodkr} - f_{jiodkr}), \qquad i \in \mathcal{P}, i \neq o, i \neq d, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{5.10}$$

$$t_{iodr} \cdot \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}(f_{ijodkr} - f_{jiodkr}) \geq 0, \qquad i \in \mathcal{P}, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{5.11}$$

Constraints (5.10-5.11) set the value of the transhipment in the transition ports for demand (o,d) on route r. The transhipment is thus associated with the route that picks up the cargo in the hub. Constraints (5.11) force the transhipment to 0 when no transhipment occurs. These constraints are not necessary in the Basic model because of the structure of the objective function - the transhipment will always get the lowest possible value. However, for some extensions of the Basic model there is a need for the transhipment being explicitly specified. For linearizations of constraints (5.11) see Section 5.3. For the sake of computational effectiveness the constraints are omitted in the computational study of the Basic model in Chapter 12.

Fleet constraints

$$\sum_{r \in \mathcal{R}} a_{kr} \leq N_k, \qquad k \in \mathcal{K} \tag{5.12}$$

$$a_{kr} \geq \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{P}} T_{ijk} \cdot x_{ijkr}, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{5.13}$$

$$\sum_{(o,d) \in \mathcal{D}} f_{ijodkr} \leq S_k \cdot x_{ijkr}, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5.14}$$

Constraints (5.12) make sure that the number of vessels in use of ship type k does not exceed the numbers of vessels in the fleet. Constraints (5.13) calculate the number of vessels of ship type k that are needed on route r. Constraints (5.14) ensure that the vessels capacity is adhered to.

Number of visits

$$b_{jk} = \sum_{i \in \mathcal{P}} \sum_{r \in \mathcal{R}} x_{ijkr}, \qquad j \in \mathcal{P}, k \in \mathcal{K} \tag{5.15}$$

Constraints (5.15) calculate the number of times each port is visited by a ship type k each week.

Subtour elimination constraints

$$\textbf{SEC}, \qquad r \in \mathcal{R} \tag{5.16}$$

The SECs are represented by (5.16). Because of their complexity these are further described separately in Section (5.2).

Binary, non-negativity and integrality constraints

$$x_{ijkr}, y_{kr} \in \{0,1\}, \qquad i,j \in \mathcal{P}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5.17}$$

$$f_{ijodkr}, t_{iodr}, a_{kr}, b_{jk} \geq 0, \qquad i,j \in \mathcal{P}, (o,d) \in \mathcal{D}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5.18}$$

$$a_{kr}, b_{jk} \qquad \text{integer}, \qquad j \in \mathcal{P}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5.19}$$

Constraints (5.17)-(5.19) impose binary, non-negativity and integrality constraints on the variables, accordingly.

5.2 Subtour elimination constraints

In this section we present the mathematical formulations of the subtour elimination constraints (SECs), represented by constraints (5.16) in the mathematical formulation.

Straightaway it may seem contradictory imposing SECs in a model where the objective is to make routes. In order to clarify this - in the LS-NDP context SECs are constraints that make sure that multiple subtours can not be assigned to the *same route index*. If we did not have these constraints the optimal solution would use less vessels than what is really possible, because routes in geographically different places could ”share” vessels. This concept is illustrated in Figure 5.2.

Making SECs for this problem differs from other routing problems, e.g. VRP, because there are no natural depots, i.e. there are no ports we know for sure that will be part of a route before we run the model. To impose SECs the model therefore has to select one port in each route to be the reference point of the route, and make sure that all other ports in the same route are linked with this reference port. The selected reference port of a route is denoted *the depot*. It does not matter which port is chosen as the depot of the route, and therefore the random selection of the depot port arises some symmetry issues. The symmetry related to the possible solutions with different depots is discussed in Section 9.1.

So, to avoid subtours we assign a variable, s_{ir}, to each port in a route representing its sequence number in the route, counted from the depot. By imposing the constraint that there cannot be an arc from one port to another port with a lower sequence number, s_{ir}, except if it is connected to the depot, we ensure that there will not be any subtours. The concept is illustrated in Figure 5.3. For instance there could not be an arc from port 5 to port 4 because $s_{51} \geq s_{41}$.

In the remainder of this section the additional variables and constraints needed to assure the elimiation of subtours are presented and explained.

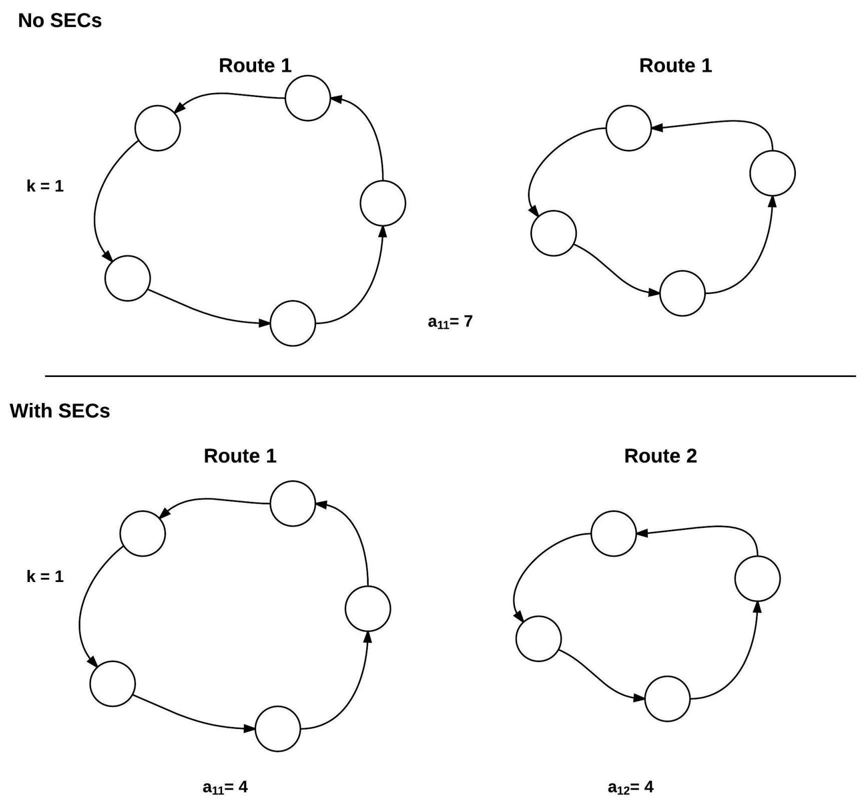


Figure 5.2: Illustration of the need for subtour elimination constraints. Without SECs two loops in geographically different places can share a vessel. This is, obviously, not possible in reality, and will result in the model reporting artificially good solutions.

Additional variables

r_{ir} - 1 if port i is visited in route r, 0 otherwise.

d_{ir} - 1 if port i is set to depot in route r, 0 otherwise.

s_{ir} - Variable that represents port i's sequence number in route r, counted from the depot.

Additional constraints

$$(\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr})\cdot(1-r_{ir}) = 0, \qquad i\in\mathcal{P}, r\in\mathcal{R} \tag{5.20}$$

$$\sum_{i\in\mathcal{P}} d_{ir} = 1, \qquad r\in\mathcal{R} \tag{5.21}$$

$$s_{jr} - s_{ir} + (N-1)\cdot(1-\sum_{k\in\mathcal{K}} x_{ijkr} + d_{jr}) \geq 1, \qquad i\in\mathcal{P}, j\in\mathcal{P}, r\in\mathcal{R} \tag{5.22}$$

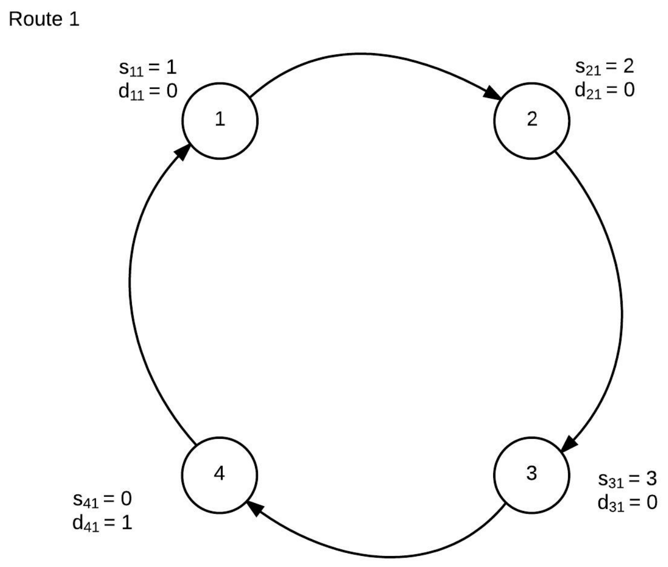


Figure 5.3: Illustration of how subtour elimination is achieved by imposing sequence variables, s_{ir}.

$$d_{ir} + \frac{1}{N-1} \cdot s_{ir} \leq 1, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{5.23}$$

$$s_{jr} - (N-1)(2 - \sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - d_{ir}) \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P}, r \in \mathcal{R} \tag{5.24}$$

Constraints (5.20) - (5.24) are all related to the elimination of subtours. Constraints (5.20) give value to the binary variable r_{ir}, that states whether or not port i is part of route r. For linearization of (5.20) see Section 5.3. Constraints (5.21) force each route to have one and only one depot. The depot serves as a reference point for the ordering of all the ports in a route in order to ensure that all of them are connected. Constraints (5.22) give each port in a route a number according to its order in the sequence starting from the chosen depot and through the route. This numbering is essential in order to avoid subtours, because it ensures that all ports must be connected to the depot through the variable s_{ir}, and since there is only one depot in each route it will only be possible to create one tour for each route. Constraints (5.23) set the depots' order s_{ir} in route r to 0. Constraints (5.24) forces the first port in each loop to have order number 1.

Binary, non-negativity and integrality constraints

$$r_{ir}, d_{ir} \in \{0,1\}, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{5.25}$$

$$s_{ir} \geq 0, \qquad \text{integer} \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{5.26}$$

Constraints (5.25)-(5.26) impose binary, non-negativity and integrality constraints on the additonal variables needed for the subtour elimination.

5.3 Linearizations

Constraints (5.11) can be linearized by substituting them by constraints (5.27-5.29):

Introduce the dummy variable c_{ijodkr}:

c_{ijodkr}	- 1 if the net flow in port i of demand (o,d) associated with route r is non-positive, 0 otherwise.

$$\sum_{k \in \mathcal{K}} (f_{ijodkr} - f_{jiodkr}) + D_{od} \cdot c_{ijodkr} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{5.27}$$

$$D_{od} \cdot (1 - c_{ijodkr}) - t_{iodr} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{5.28}$$

$$c_{ijodkr} \in \{0,1\}, \qquad i,j \in \mathcal{P}, (o,d) \in \mathcal{D}, k \in \mathcal{K}, r \in \mathcal{R} \tag{5.29}$$

Constraints (5.27) set the value of c_{ijodkr} to 1 when the net flow in port i of demand (o,d) associated with route r is non-positive, i.e. there are no transhipment. Finally, constraints (5.28) enforce the transhipment to 0 when there is no transhipment.

Constraints (5.20) can be linearized by replacing them with constraints (5.30)-(5.31):

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - r_{ir} \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{5.30}$$

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - r_{ir} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{5.31}$$

Constraints (5.30) set variable r_{ir} to 1 if port i is visited by route r, while constraints (5.31) force r_{ir} to 0 if port i is not visited by route r.

Chapter 6

Flower model

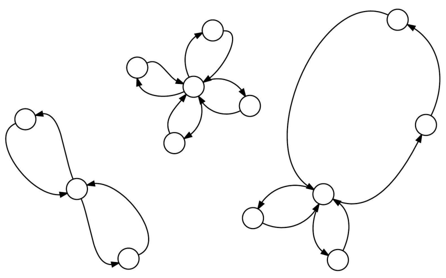


Figure 6.1: Examples of flower routes. All the routes above are enabled with the *Flower model*

.

The first formulation that incorporates complex route structures is called the *Flower-model*, because it allows a route to sail in a "flowery-pattern". The model enables routes where one and only one port in a route to be visited an indefinite number of times. The port that is visited multiple times is denoted *the butterhub* of a route. Examples of Flower routes can be seen in Figure 6.1.

In this section we present the properties of the Flower model, along with an explanation of the main conceptual challenges that arise with the modelling. In Section 6.2 the mathematical formulation is provided.

The Flower model does not require any changes to the underlying structure from the Basic model, i.e. we can keep the same sets and indices. However, there is a need for a considerable amount of new variables and constraints.

6.1 Enabling the Flower structure

There are two main issues related to the modeling of the Flower model; facilitating the flower structure and the calculation of internal transhipment. In this section we will describe the conceptual issues in words, while the additional mathematical formulation is presented later in Section 6.2.

1. **Facilitating the flower routes.**

   In the Basic model only simple routes are permitted. In the Flower model, on the other hand, we want a route to be able to contain several connected loops. To allow this structure two actions are required:

   (a) Allow a port to have multiple arcs coming in and out from the same route.

   (b) Make sure that the butterhub is set to be the depot of the flower route.

   When the butterhub is set to be the depot its sequence variable s_{ir} will be set to 0 and this way we ensure that loops are allowed as long as they are connected to the butterhub. Recall that there can be only one depot of a route. This means that there can also only be one butterhub in the flower-route, which again imposes the elimination of "subflowers". Figure 6.2 illustrates how the flower structure is allowed when the two above criteria are met.

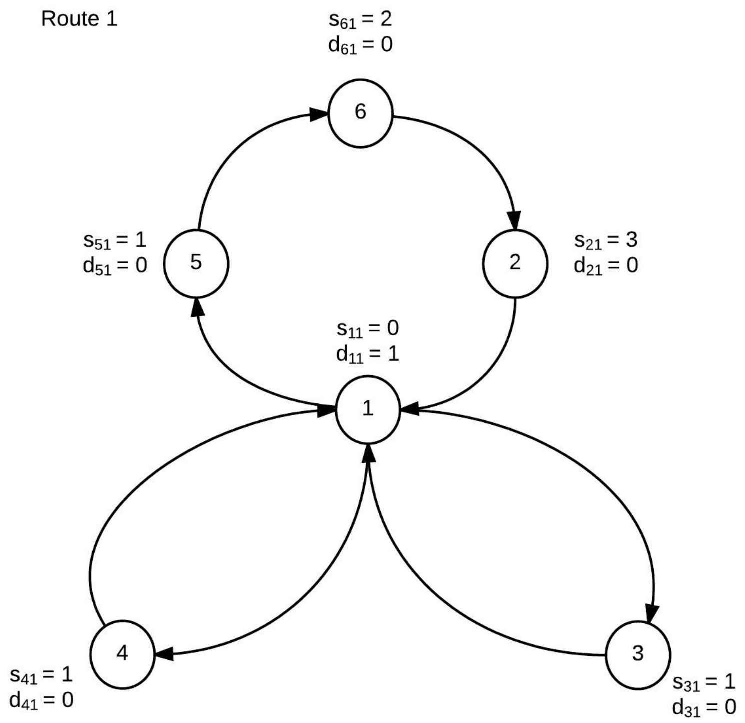


Figure 6.2: Facilitating the flower structure: Illustration of how several loops are allowed when the butterhub is set to be the depot of the route and ports can have several incoming and outgoing arcs.

2. **Capturing the internal transhipment.**

   Recall from Chapter 1 that butterfly routes open up the possibility for a route to tranship to another ship in the same route, and thus socalled *internal transhipment* can occur. Figure 6.3 shows an example where internal transhipment takes place: The cargo is dropped off in port 1, i.e. the butterhub, and later picked up by the same route before it gets transported to its final destination port 2. Since the internal transhipment does not affect the net flow of a demand through a butterhub, as opposed to the external transhipment, we need some way of examining how the flow changes between the different loops of the flower. To do this we need a way to separate the different loops from each other and in which order they are in the route. The calculation of the internal transhipment in the Flower approach is quite intricate, and it requires a significant amount of new variables and constraints. The calculations of the internal transhipment is thoroughly explained in Section 6.2.

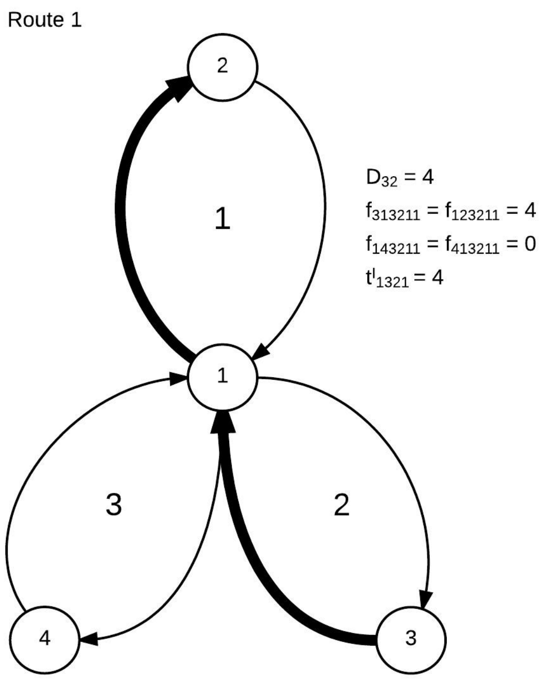


Figure 6.3: Example of internal transhipment. Cargo is dropped off in the butterhub and picked up later by the same route.

6.2 Flower formulation

In this section we present the modifications and additions we need in the mathematical formulation of the Basic model in order to obtain the desired properties of the Flower model. A complete compact version of the Flower model can be found in Appendix A.3.

Modifications We have to substitute the transhipment variable from the Basic model, t_{iodr}, into internal and external transhipment, t^{I}_{iodr} and t^{E}_{iodr}, respectively.

$$t_{iodr} \quad \rightarrow \quad t^{E}_{iodr} + t^{I}_{iodr}$$

t^{I}_{iodr}	- Internal transhipment of demand (o, d) in butterhub i of butterfly route r
t^{E}_{iodr}	- External transhipment of demand (o, d) in port i of route r

Due to the substitution of t_{iodr} into t^{I}_{iodr} and t^{E}_{iodr}, the modified objective function becomes:

$$minimize \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} C^{S}_{k}\cdot\ a_{kr} + \sum_{i\in\mathcal{P}}\sum_{o\in\mathcal{P}}\sum_{d\in\mathcal{P}}\sum_{r\in\mathcal{R}} C^{T}_{i}\cdot(t^{E}_{iodr}+t^{I}_{iodr}) + \sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}} C^{P}_{ik}\cdot\ b_{ik} \tag{6.1}$$

Additional variables

m_{ir}	- 1 if port i has multiple arcs (i.e. is a butterhub) in route r, 0 otherwise
n_{ir}	- The loop number of port i in route r
l_{r}	- The number of loops in route r
e^{F}_{ir}/e^{L}_{ir}	- 1 if port i is in the first/last loop (n_{ir}=1/n_{ir}=l_r), 0 otherwise
g^{N}_{ijr}/g^{S}_{ijr}	- 1 if port i's loop/order number is greater than port j's, 0 otherwise
h_{ijr}	- 1 if port i and port j are in the same loop ($n_{ir} = n_{jr}$), 0 otherwise
o_{ijr}	- Difference between port i and port j's loop number (n_{ir}-n_{jr})
p^{S}_{ijr}	- 1 if port i is in the last loop (e^{L}_{ir}=1) and port j is in the first loop (e^{F}_{jr}=1)
p^{N}_{ijr}	- 1 if port i and j are in neighbouring loops and port i's loop is sailed before port j's loop
q_{imodr}	- The positive pairwise difference between out-flow and in-flow of demand (o, d) in neighbouring loops in Flower route r

Enabling Flower routes

Modifications in constraints

Constraints (5.30) from the subtour elimination in the basic model have to be replaced by Constraints (6.2) because a port now can have several arcs coming out belonging to the same route.

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - 2(N-1) \cdot r_{ir} \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.2}$$

Construction of the additional variables

As described above the flower-approach requires a substantial number of new variables to enable butterfly routes. Below the constraints needed to construct the new variables are presented:

m_{ir}*- 1 if port* i *is butterhub in route* r*, 0 otherwise*

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - (N-1) \cdot m_{ir} \leq 1, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.3}$$

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - 2 \cdot m_{ir} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.4}$$

$$m_{ir} - d_{ir} \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.5}$$

Constraints (6.3)-(6.4) set the value of m_{ir} to 1 if port i has more than 2 arcs, 0 otherwise. Constraints (6.5) ensure that if a port i has more than two connecting arcs in a route it should be set to the depot of the route ($m_{ir} = 1 \implies d_{ir} = 1$).

Calculations of internal transhipment

l_r *- The number of loops in route* r

$$l_r \leq \sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} + (N-1)(1 - d_{ir}), \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.6}$$

Constraints (6.6) give value to the variable l_r that states how many loops there are in a route. This is done by calculating how many arcs that are going into the butterhub, and stating that l_r cannot be greater or less than this value. In a non-butterfly route the value of l_r is 1. Together with constraints (6.11) they ensure that variable l_r only can have the value equal to the exact number of loops in a route.

n_{ir} - *The loop number of port i in route r*

$$n_{ir} - (N-1)(1-d_{ir}) \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.7}$$

$$n_{ir} - n_{jr} - (N-1)(1 - \sum_{k \in \mathcal{K}} (x_{ijkr} + x_{jikr}) + 2d_{ir} + 2d_{jr}) \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.8}$$

$$|(N-1)(s_{ir} - s_{jr}) + n_{ir} - n_{jr}| \geq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.9}$$

$$s_{ir} + d_{ir} + (1 - y_{kr}) \geq 1, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.10}$$

$$n_{ir} \leq l_r, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.11}$$

Constraints (6.7)-(6.11) above are related to variable n_{ir} and the *loop numbers* of the butterfly-route. They make sure that the butterhub gets loop number 0 (n_{ir}=0), that ports in the same loop have the same loop number and that ports in different loops have different loop numbers and that the loops are numbered from 1 to the total number of loops in the route (l_r). This is needed because we need to know the sequence of which the different loops are sailed, in order to calculate the internal transhipment.

Constraints (6.7) set the loop number, n_{ir}, of the butterhub to 0. Constraints (6.8) make sure that ports in the same loop get the same loop number, by stating that if there is an arc between two ports in the same route, none of them being the depot port of the route, they must have equal loop numbers. Constraints (6.9) make sure that ports in different loops in route r get different loop numbers. This is done by stating that ports in a route with equal order number, s_{ir}, have to have different loop numbers. The linearization of these constraints are explained later in the paper, and can be replaced by constraints (6.26). Constraints (6.10) ensure that non-butterhubports have to have loop numbers greater or equal to 1, while constraints (6.11) ensure that a loop number cannot be greater than the total number of loops in a route.

e^F_{ir}/e^L_{ir} *- 1 if port i is in the first/last loop, 0 otherwise*

$$n_{ir} - 2(1 - e^F_{ir} - d_{ir}) \geq 0 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.12}$$

$$n_{ir} - (N-1)(1 - e^F_{ir}) \leq 1 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.13}$$

$$e^F_{ir} - n_{ir} \leq 0 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.14}$$

$$e^L_{ir} - n_{ir} + l_r \geq 1 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.15}$$

$$n_{ir} - l_r + (N-1)(1 - e^L_{ir}) \geq 0 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.16}$$

Constraints (6.12-6.16) give values to the binary variables e^F_{ir} and e^L_{ir} that state whether or not a port is in the first or last loop, respectively. Constraints (6.14) ensure that the variable e^F_{ir} is set to 0 for the butterhub, while e^L_{ir} is automatically set to 0 for the butterhub by constraints (6.16).

g^N_{ijr}/g^S_{ijr} *- 1 if port i's loop/order number is greater than port j's, 0 otherwise :*

$$n_{ir} - n_{jr} - (N-1)g^N_{ijr} \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.17}$$

$$n_{ir} - n_{jr} + (N-1)(1 - g^N_{ijr}) \geq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.18}$$

$$s_{ir} - s_{jr} - (N-1)g^S_{ijr} \leq 0 \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.19}$$

$$s_{ir} - s_{jr} + (N-1)(1 - g^S_{ijr}) \geq 1 \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.20}$$

Constraints (6.17)-(6.20) set the value of the binary variables g^N_{ijr}/g^S_{ijr} to 1 if port i's loop/order number, respectively, is greater than port j's loop/order number, and to 0 otherwise. Constraints (6.17)-(6.18) ensure correct value of variable g^N_{ijr}, while constraints (6.19)-(6.20) are related to variable g^S_{ijr}. Constraints (6.17) and (6.19) enforce 1, while constraints (6.18) and (6.20) enforce 0 on the binary variables.

h_{ijr} *- 1 if ports i and j are in the same loop, 0 otherwise*

$$g^N_{ijr} + g^N_{jir} + h_{ijr} = 1 \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.21}$$

Constraints (6.21) set the binary variable h_{ijr} to 1 if ports i and j are in the same loop, 0 otherwise. Constraints (6.21) state that if both variables g^N_{ijr} and g^N_{jrr} are 0 (which means that none of port i or port j has greater loop number than the other port, i.e. they have the same loop number), h_{ijr} is set to 1. 6.21 also state that if g^N_{ijr} or g^N_{jir} is 1, h_{ijr} is set to 0.

o_{ijr} *- the positive difference between port i and j's loop numbers*

$$o_{ijr} - n_{ir} + n_{jr} + (N-1)(1 - g^N_{ijr}) \geq 0, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.22}$$

$$o_{ijr} - n_{ir} + n_{jr} - (N-1)(1 - g^N_{ijr}) \leq 0, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.23}$$

$$o_{ijr} - (N-1)(1 - g^N_{jir}) \leq 0, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.24}$$

$$o_{ijr} - (N-1)(1 - h_{ijr}) \leq 0, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.25}$$

$$o_{ijr} + o_{jir} + (N-1)h_{ijr} \geq 1, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.26}$$

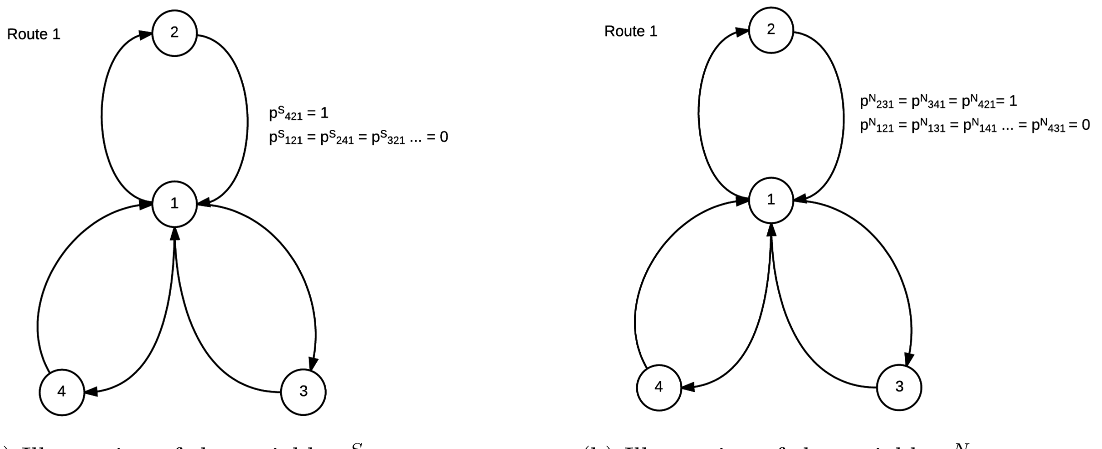
Constraints (6.22)-(6.26) together set the value of the variable o_{ijr} to max $\{n_{ir}$-n_{jr},0$\}$. Constraints (6.22)-(6.23) set the value of o_{ijr} to the difference between ports i and j's loop number if port i's loop number is greater than the loop number of port j. Constraints (6.24) set o_{ijr} to 0 if port i's loop number is less than port j's loop number. Constraints (6.25) set o_{ijr} to 0 when ports i and j are in the same loop, while constraints (6.26) ensure that ports in different loops get different loop numbers.

p^S_{ijr} - *1 if port i is in the last loop and j in the first loop of route r*

$$e^L_{ir} + e^F_{jr} - 2p^S_{ijr} \leq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.27}$$

$$e^L_{ir} + e^F_{jr} - 2p^S_{ijr} \geq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.28}$$

Constraints (6.27-6.28) set the value of the variable p^S_{ijr} to 1 if port i is in the last loop and j is in the first loop, and 0 otherwise. An illustration of the variable p^S_{ijr} can be seen in Figure 6.4.


Route 1
2
1
4
3
$p^S_{421} = 1$
$p^S_{121} = p^S_{241} = p^S_{321} ... = 0$
Route 1
2
1
4
3
$p^N_{231} = p^N_{341} = p^N_{421} = 1$
$p^N_{121} = p^N_{131} = p^N_{141} ... = p^N_{431} = 0$

(a) Illustration of the variable p^S_{ijr} (b) Illustration of the variable p^N_{ijr}

Figure 6.4: Illustration of the variables p^S_{ijr} and p^N_{ijr}. **(a)** The variable p^S_{ijr} is 1 if port i is in the last loop and port j in the first loop, 0 otherwise. **(b)** Variable p^N_{ijr} is 1 if port i and j are in neighbouring loops and port i's loop is sailed before port j's loop, 0 otherwise. ($p^S_{ijr} = 1 \implies p^N_{ijr} = 1$).

p^N_{ijr} - *1 if ports i and j are in neighbour loops and route r sails j's loop directly after i's loop.*

$$p^S_{ijr} - p^N_{ijr} \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{6.29}$$

$$o_{jir} + 2g^N_{ijr} + 2h_{ijr} + p^N_{ijr} + 2d_{ir} + 2d_{jr} \geq 2, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.30}$$

$$o_{ijr} + o_{jir} - (N-1)(1 - p^N_{ijr} + p^S_{ijr}) \leq 1, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.31}$$

$$p^N_{ijr} + h_{ijr} \leq 1, \qquad i, j \in \mathcal{P}, r \in \mathcal{R} \tag{6.32}$$

Constraints (6.29)-(6.32) set the value of the variable p^N_{ijr} to 1 if port i and j are in neighbour loops and i's loop is sailed before j's loop, 0 otherwise. Constraints (6.29)-(6.30) ensure that p^N_{ijr} is 1 if $o_{jir} = 1$, i.e. ports i and j are neighbour loops and port i's loop is sailed before port j's loop. Constraints (6.31)-(6.32) set p^N_{ijr} to 0 if i and j are not in neighbour loops. See Figure 6.4 for illustration of variable p^N_{ijr}.

The one additional "help-variable" q_{imodr}, which is the positive pairwise difference between out-flow and in-flow of demand (o, d) in neighbouring loops in route r, is illustrated in Figure 6.5.

$$q_{imodr} \geq \sum_{k \in \mathcal{K}} (f_{jmodkr} - f_{ijodkr} - D_{od}(4 - m_{jr} - p^N_{ijr} - x_{ijkr} - x_{jmkr}), \qquad (i,j),(j,m) \in \mathcal{A}, r \in \mathcal{R} \tag{6.33}$$

Constraints (6.33) calculate the positive difference between the flow coming from the depot into a loop and the flow coming into the depot coming in from the loop that is sailed before. If this value is positive this means that something is picked up in the depot. It only makes sense to calculate q_{imodr} when: r is a butterfly route with butterhub j (m_{jr}=1), port i and port m are in neighbouring loops and i is in the loop before m (p^N_{ijr}=1), and there is a direct connection between the butterhub j and i and m ($x_{ijkr} = x_{jmkr} = 1$).

$$t^I_{jodr} \geq \sum_{i \in \mathcal{P}} \sum_{m \in \mathcal{P}} q_{imodr} - t^E_{jodr}, \qquad j \in \mathcal{P}, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{6.34}$$

Constraints (6.34) calculate the internal transhipment in the butterhub by summarizing all positive pairwise differences (q_{imodr}) and subtracting the external transhipment t^E_{jodr}. The

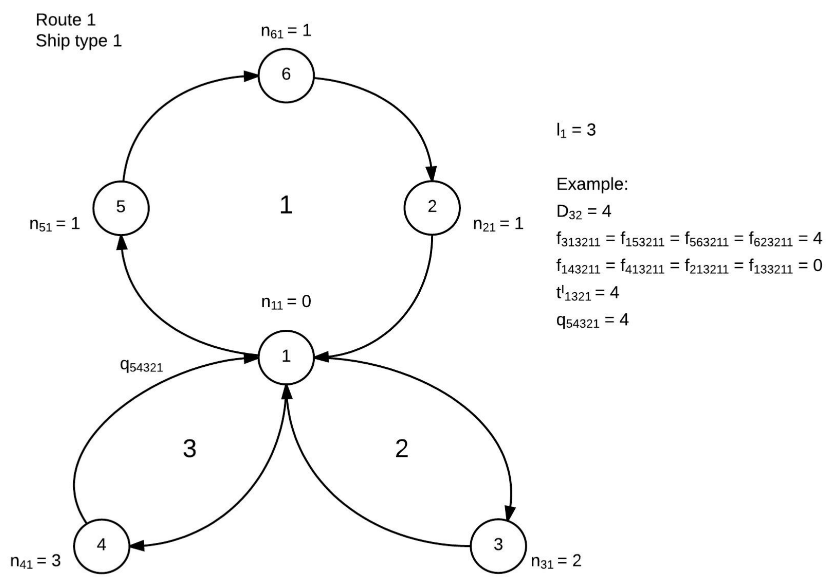


Figure 6.5: Illustration of the variable q_{imodr}. We see that the incoming flow of demand (3,2) from port 4 to the butterhub, port 1 (f_{413211}), is zero, while the successive outgoing flow from the butterhub to port 5 (f_{153211}) is 4. This leads to a *positive pairwise difference* of 4 which means that 4 units of demand (3,4) has been picked up in the butterhub between loop 3 and loop 1. Since there is only one route in this network we know that all positive pairwise differences are caused by internal transhipment.

rationale as to why the internal transhipment can be calculated this way is the following: when the variable is positive this means that a vessel has picked up something in the depot that it did not bring. The demand that is picked up can be one of two things: external transhipment *or* internal transhipment. Thus, by summarizing all positive pairwise differences and exploit the fact that we already know the value of the external transhipment; what's left has to be the internal transhipment.

Binary, non-negativity and integrality constraints

$$m_{ir}, e^F_{ir}, e^L_{ir}, g^N_{ijr}, g^S_{ijr}, h_{ijr}, p^S_{ijr}, p^N_{ijr} \in \{0,1\}, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.35}$$

$$n_{ir}, o_{ir}, l_r, q_{imodr} \geq 0, \qquad \text{integer} \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{6.36}$$

Constraints (6.35-6.36) impose binary, non-negativity and integrality constraints on the additonal variables needed to allow butterfly routes with the flower approach.

Chapter 7

Chain model

In this chapter we present another formulation to enable complex route structures than the one presented in Chapter 6.

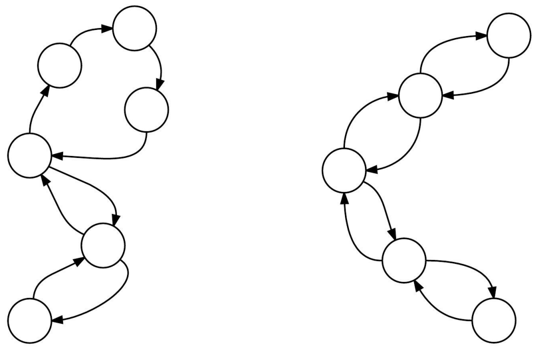


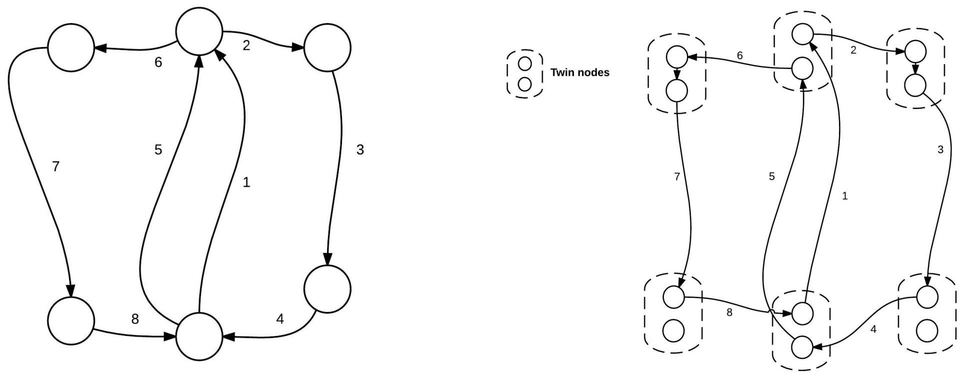
Figure 7.1: Chain routes. The Chain model allows a route to visit multiple ports twice.

The two models are fundamentally different, both in which types of complex route structures they allow, and also how they are formulated.

The model presented in this chapter is called the *Chain model* because it facilitates the modelling of "chain-like"-routes. Figure 7.1 shows examples of routes that can be constructed with the Chain model.

The Chain model allows a route to visit several ports more than once. However, each port can only be visited at most twice. It enables a route to traverse the same arc twice. An arc that is traversed twice is denoted a *butterarc*. Figure 7.2 shows an example of a butterarc and how it can be realized with the Chain model.

A virtual double network, called the *twin network*, forms the foundation of the formulation of the Chain model. The properties and advantages of this network is explained in Section 7.1. In Section 7.2 the mathematical formulation of the Chain model is given.



(a) Butter arc (b) Butter arc realization in a twin network

Figure 7.2: Figure 7.2a shows an example of a *butterarc* - an arc that is traversed twice in the same route. Figure 7.2b shows how the butterarc can be realized in a twin network. The Flower model does not facilitate butter arcs.

7.1 Twin network

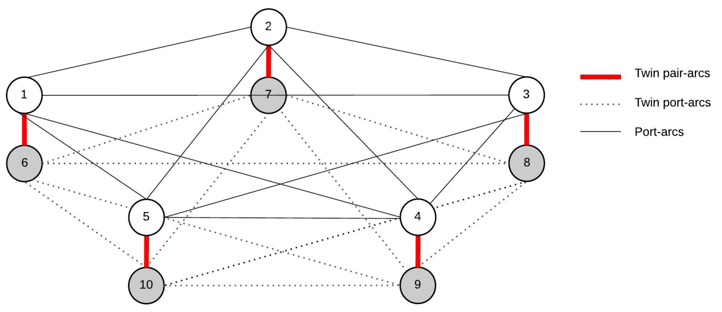
The basis of the Chain model is a virtual double network - called the *twin network*. Figure 7.3 illustrates the concept of the twin network used in the Chain model, and shows how complex route structures can be enabled by utilizing the network.

In the twin network all the ports are duplicated, each port getting an associated twin port. Doubling the network makes it possible to visit the same port twice by visiting the original port *and* its twin, and count this as two visits to the same port. This way, we do not have to alter the flow constraints that apply for each node in the network.

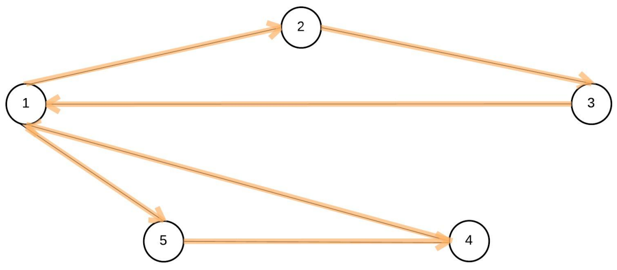
The set of original ports and twin ports is called nodes and denoted $\mathcal{N}$. Original ports are indexed from 1 to N, while the twin ports are indexed from $N+1$ to $2N$. Port i's twin is indexed $i+N$. This makes it possible to refer to the original port and its twin port together.

The twin network has the following properties:

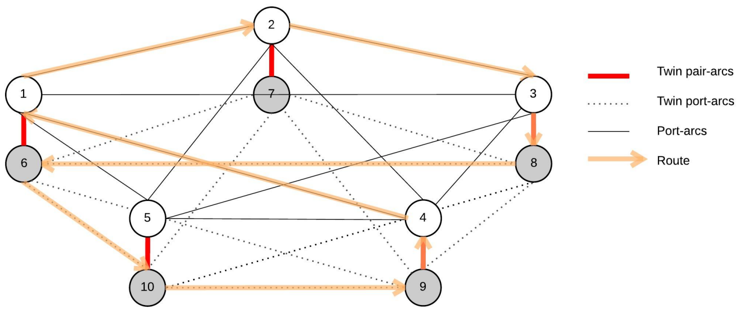
- The distance between port i and its twin port $i+N$ is 0, i.e. $T_{i,i+N} = T_{i+N,i} = 0$.
- The distance between twin port $i+N$ and twin port $j+N$ is equal to the distance between port i and j, i.e. $T_{i+N,j+N} = T_{ij}$.
- The transhipment cost of twin port $i+N$ is equal to the transhipment cost of port i, i.e. $C^T_{i+N} = C^T_i$. The same applies for the visiting cost.



(a) Twin network



(b) Butterfly route



(c) Twin network with a route

Figure 7.3: Twin network. Figure 7.3a shows the twin network structure in the Chain model. All ports are duplicated getting a corresponding twin node. The white nodes represent the original ports while the grey nodes represent the twin ports. Figure 7.3b shows a butterfly route, and Figure 7.3c shows how this butterfly route is realized in the Chain network structure.

7.2 Mathematical formulation of the Chain model

In this section the additional notation and changes necessary in the Basic model to enable the described properties of the Chain model.

Unlike the Flower model, the realization of the Chain model requires few new variables and constraint formulations. On the other hand, in the Chain model we need a new set, and make some assumptions regarding the combinations of indices of which the variables are defined.

Sets and indices

$$\mathcal{N} \quad \text{- Nodes, } i,j \qquad i,j \in \{1..2N\}$$

We introduce the set of nodes $\mathcal{N}$ which consists of both original ports and twin ports. The original ports are indexed from 1 to N, while the twin ports are indexed from $N+1$ to $2N$. Port i's twin is indexed $i+N$. With regards to the set of arcs $\mathcal{A}$ this is not defined for any tuples of ports and twin ports, except between a port and its own twin, i.e. $(i,i+N)$.

Constraints

Number of visits

We have to make a slight change in constraints (5.15) from the Basic model because a move between a port and its twin port should not be counted as a visit. This is because they are in reality the same port, and we have to avoid double counting the visits. So, constraints (5.15) have to be modified to:

$$b_{ik} = \sum_{j\in\mathcal{P}, j\neq i+N,} \sum_{r\in\mathcal{R}} x_{ijkr}, \qquad i\in\mathcal{P}, k\in\mathcal{K} \tag{7.1}$$

Demand flow

It is not necessary to do any changes in regards to the ship flow x_{ijkr} - we still need ship flow conservation in every node. On the other hand, when it comes to the demand flow f_{ijodkr} we have to do some changes. This is to enable transhipment in all ports. If one route visits port i and leaves some units of demand (o,d) we want a route that visits port i's twin port, port $i+N$, to be able to pick up these units without visiting port i. In order to allow this we redefine the demand flow conservation in a node to be a joint demand flow conservation for twin ports. This means that the net flow into port i and port $i+N$ has to be equal to the net flow going out of the nodes, but not necessarily for each node. The same concept applies in origin ports and destination ports the joint outgoing/incoming flow has to equal the demand (o,d).

Therefore we have to substitute constraints (5.6-5.8) from the Basic model with the corresponding constraints (7.2-7.4).

$$\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(f_{ijodkr} + f_{i+N,jodkr} - f_{jiodkr} - f_{j,i+N,odkr}) = 0, \qquad j \in \mathcal{N}, (o,d) \in \mathcal{D}, j \neq o, j \neq d, j \leq N \tag{7.2}$$

$$\sum_{j\in\mathcal{P},j\neq o,j\neq o+N}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(f_{ojodkr} + f_{o+N,jodkr}) = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{7.3}$$

$$\sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(f_{idodkr} + f_{i,d+N,odkr}) = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{7.4}$$

Calculation of internal transhipment

In the Chain model the calculation of the internal transhipment does not require any extra consideration. The internal transhipment is taken care of by constraints (5.10) because we, unlike what we do with the demand flow conservation constraints do not calculate the flow balance for the port and its twin port in total. Thus, constraints (5.10) will track a change in the flow for each node separately, hence discovering any transhipment that happens in each node.

Chapter 8

Transit time extensions

As has already been mentioned, offering short transit time is a major competitive factor for liner shipping companies, especially when the goods transported are time sensitive. Also, liner shipping networks that are optimized only with respect to cost get an unrealistically high network utilization as containers are allowed on detours that offer unused capacity but in practice they will violate transit time restrictions (Karsten et al., 2015). By introducing a maximum transit time for each commodity in the network, the number of allowed paths will be limited significantly for the individual commodities and introduces new limits on the feasible solutions in the network design process (Karsten et al., 2015).

In this chapter we present the formulations that extend the Basic, Flower and Chain models to take transit time into account. From here the models without transit time will be collectively denoted the *original* models, while the models incorporating transit time are called the *transit time* models.

The way the transit time requirements are incorporated is equal for all models. Section 8.1 cover some of the assumptions and modeling choices done with regards to the incorporation of transit time in the models. In Section 8.2-8.4 the mathematical formulations of the transit time extensions of the Basic, Flower and Chain models, are presented.

8.1 Assumptions and modeling choices

The transit time models make it possible to specify a maximum transit time for each *order*. By order we mean a bundle of containers with the same origin and destination port. How many containers that constitute an order is decided by the user. The transit time models construct routes that ensure that all containers are delivered within their specified transit time requirement. Between one single O-D pair, there can be several orders with different transit times. This is illustrated in Figure 8.1. The orders can take different paths depending on their transit time requirement. The Figure illustrates that even though there are two orders going from O to D, they take different paths. This can for instance be because there is not enough capacity on the shortest path.

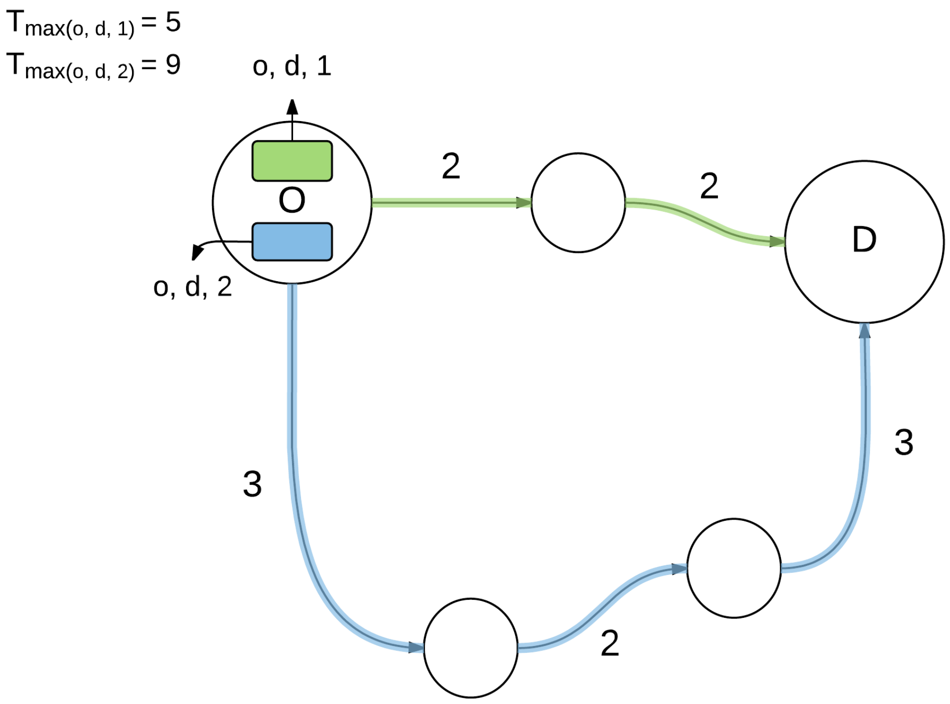


Figure 8.1: Transit time. The figure illustrates that different orders with the same origin and destination can take different paths because of their individual transit time requirement. The green order (o,d,1) could not have taken the blue path, and still comply with the transit time requirement. The blue order (o,d,2) could have taken the green path, but there may be other reasons for not doing it, for instance the capacity on the path is not sufficient.

Enabling the properties described above requires some changes from the original models. Two new sets are introduced; the set Ω is the set of possible order indices available for orders between a particular O-D pair. The second set is the set of orders, $\mathcal{O}$, which consists of all order triplets (o, d, ω). Thus, compared with the original models, the tuplet (o, d) is replaced by the triplet (o, d, ω). This way it is possible to refer to a particular order. The size of each order is given by the parameter $D_{od\omega}$.

Binary restrictions are imposed on the flow variable $f_{ijod\omega kr}$. The binary restrictions on the flow variable imply that the containers included in the same order have to follow the same path. The reason for this is that if each order could be spilt, calculating the transit time of all paths of the parts of the order would make the complexity of the model excessive. An implication of the binary restrictions imposed on the flow variable, is that the order sizes will affect the demand splitting possibilities. However, since the sizes of the demands are parameters specified by the user, this provides an opportunity rather than a limitation.

Transit time is defined as the time from when a container leaves its origin port until it arrives its destination port. The models assume that the transit time consists of two parts; *Sailing time* and *Transhipment time*, as can be sees from (8.1). It could be argued that time in port also should be included, but we consider this as a part of the sailing time.

$$\textbf{Transit time}_{od\omega} = \text{Sailing time}_{od\omega} + \text{Transhipment time}_{od\omega}, \quad (o,d,\omega) \in \mathcal{O} \tag{8.1}$$

The sailing time is the time the container order is at sea sailing from one port to another. The calculation of the sailing time is given by (8.2). It shows that the sailing time is calculated by multiplying each arc (i,j) where order (o,d,ω) is transported with the time T_{ijk} that ship type k needs to sail it. It should be noted that this is a slight simplification, because the vessels seldom sail at full speed because of the weekly frequency requirement. So if route r takes 3.6 weeks to sail the vessel still needs to spend 4 weeks on it. In reality this is the same as if the vessel had a speed that is $\frac{4-3.6}{3.6} \approx 11.4$ % less than their real speed. Thus, the sailing time will be lower than what it is in reality. There are many ways to compensate for this. However, we do not think this will affect the solution too much and we have therefore chosen to stay with the simplification shown in (8.2).

$$\textbf{Sailing time}_{od\omega} = \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} T_{ijk}\cdot f_{ijod\omega kr}, \quad (o,d,\omega)\in\mathcal{O} \tag{8.2}$$

The transhipment denotes the time an order that is being transhipped has to wait in the hub port before it gets picked up. This is extremely complicated to calculate exact and we have therefore made the simplification that the transhipment time always is half a week. This serves as a good approximation because we know that the longest time possible it has to wait is one week, while the lowest time is that it gets picked up immediately. This follows from the weekly frequency requirement.

The transhipment time can thus be written:

$$\textbf{Transhipment time}_{od\omega} = \sum_{i\in\mathcal{P}}\sum_{r\in\mathcal{R}} \frac{t_{iod\omega r}}{2}, \quad (od\omega)\in\mathcal{O} \tag{8.3}$$

8.2 Basic model with transit time

In this section the additional formulation needed in order to extend the Basic model to take into account transit time, is presented.

Additions and changes in sets and indices

Ω	- The set of possible order indices per O-D pair,	ω
$\mathcal{O}$	- Orders, (o,d,ω)	$o, d \in \mathcal{P}, \omega \in \Omega$

The set of demands $\mathcal{D}$ in the original models is substituted by the set of orders, $\mathcal{O}$. Thus all tuplets (o,d) are replaced by the triplets (o,d,ω).

Additions and changes in parameters

$D^B_{od\omega}$	- 1 if there is demand of order ω from port o to port d, 0 otherwise	$(o, d, \omega) \in \mathcal{O}$
$D^Q_{od\omega}$	- Weekly quantity (in TEU) of order (o,d,ω)	$(o, d, \omega) \in \mathcal{O}$
$T^T_{od\omega}$	- Transit time requirement (in weeks) for order (o,d,ω)	$(o, d, \omega) \in \mathcal{O}$

The parameters $D^B_{od\omega}$ and $T^T_{od\omega}$ are added to the model. The demand quantity parameter, $D^Q_{od\omega}$, is redefined to be per order (o,d,ω), and not per O-D pair as in the original model.

Changes in variables

$f_{ijod\omega kr}$	- 1 if demand (o,d,ω) is transported by a vessel of type k on arc (i,j) in route r, 0 otherwise
$t_{iod\omega r}$	- 1 if demand (o,d,ω) is transhipped to route r in port i, 0 otherwise

There is no need for additional variables, however the flow and transhipment, $f_{ijod\omega kr}$ and $t_{iod\omega r}$, variables need to be redefined. Both variables need the additional index, ω. We also have to impose binary requirements on them. This is because each order has to follow the same O-D path. Note that this does not mean that the all cargo from O-D have to follow the same path, only the cargo within the same order.

Transit time constraints

Besides imposing transit time constraints no additional constraints are needed. The Transit time constraints (TTCs) ensure that all orders are transported from their origin to their destination within the specified transit time requirement, $T^T_{od\omega}$.

The transit time constraints are given by:

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(2\cdot T_{ijk}\cdot f_{ijod\omega kr}+t_{iod\omega r})\leq 2\cdot T^{T}_{od\omega},\qquad (o,d,\omega)\in\mathcal{O}\tag{8.4}$$

Constraints (8.4) ensure that all demands (o,d,ω) are transported from their origin port o to their destination port d so that their transit time requirement, $T^{T}_{od\omega}$, is met.

8.3 Flower model with transit time

In this section the additional formulation needed in order construct the Flower transit time model (Flower TT), is presented. The same additions and changes required to obtain the Basic transit time model from the Basic model, also applies when including transit time in the Flower model. There are also some modifications to the Flower specific constraints and variables. These are listed below.

$t^{I}_{iod\omega r}$	- Internal transhipment of order (o,d,ω) in butterhub i of butterfly route r
$t^{E}_{iod\omega r}$	- External transhipment of order (o,d,ω) in port i of route r
$c_{ijod\omega kr}$	- 1 if the net flow in port i of order (o,d,ω) associated with ship type k and route r is non-positive, 0 otherwise.
$q_{imod\omega r}$	- The positive pairwise difference between out-flow and in-flow of order (o,d,ω) in neighbouring loops in butterfly route r

In constraints (6.33), the flow variables are now binary and need to be multiplied with the order size to correctly set the pairwise positive difference.

$$q_{imod\omega r}\geq\sum_{k\in\mathcal{K}}(D_{od\omega}(f_{jmod\omega kr}-f_{ijod\omega kr})-D_{od\omega}(4-m_{jr}-p^{N}_{ijr}-x_{ijkr}-x_{jmkr}),\qquad (i,j),(j,m)\in\mathcal{A},r\in\mathcal{R}\tag{8.5}$$

The same applies for the calculation of external and internal transhipment. Constraints (5.10) and (6.34) have to be substituted with (8.6) and (8.7), respectively.

$$t^{E}_{iod\omega r}\geq\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}D_{od\omega}(f_{ijodkr}-f_{jiodkr}),\qquad i\in\mathcal{P},i\neq o,i\neq d,(o,d,\omega)\in\mathcal{O},r\in\mathcal{R}\tag{8.6}$$

$$t^I_{iod\omega r} \geq \sum_{j\in\mathcal{P}} \sum_{m\in\mathcal{P}} q_{jmod\omega r} - D_{od\omega}(1-d_{ir}) - t^E_{iod\omega r}, \qquad i \in \mathcal{P}, (o,d,\omega) \in \mathcal{O}, r \in \mathcal{R} \tag{8.7}$$

The transit time constraints in the Flower TT model become:

$$\sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{P}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} (2\cdot T_{ijk} \cdot f_{ijod\omega kr} + t^I_{iod\omega r} + t^E_{iod\omega r}) \leq 2\cdot T^T_{od\omega}, \qquad (o,d,\omega) \in \mathcal{O} \tag{8.8}$$

8.4 Chain model with transit time

In this section the additional formulation needed in order construct the Chain transit time model (Chain TT), is presented. The same additions and changes in the Basic model also applies for the Chain model. The only difference in the realization of transit time requirements in the models is that the Chain model has to take into account the two layers in the transit time constraints:

$$2\cdot T_{ijk} \cdot (\sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{P}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} f_{ijod\omega kr} + \sum_{i\in\mathcal{N}\cap\bar{\mathcal{P}}} \sum_{j\in\mathcal{N}\cap\bar{\mathcal{P}}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} f_{ijod\omega kr}) + \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}} t_{iod\omega r}) \leq 2\cdot T^T_{od\omega}, \qquad (o,d,\omega) \in \mathcal{O} \tag{8.9}$$

Constraints (8.9) ensure that all orders (o,d,ω) are transported from their origin port o to their destination port d so that their transit time requirement, $T^T_{od\omega}$, is met. In the Chain model it is important to ensure that it is irrelevant if a demand arrives in a port or its twin port with regards to the transit time.

Chapter 9

Strengthening formulations

In this chapter we present different possibilities of strengthening the formulations of the three models explained in the previous chapters. First, in Section 9.1 we present measures trying to remove or reduce the symmetry in the three models. After this, in Section 9.2, valid equalities that can be added to the formulations, are introduced. At last, in Section 9.3 we show how the valid inequalities can be further enhanced with integer rounding.

9.1 Symmetry breaking constraints

Symmetry is a common problem in many combinatorial problems. The term symmetry means solutions that are different mathematical solutions, but that represent the same practical solutions. Symmetries increase the size of the search space. We must take into account symmetry or we will waste much time visiting symmetric solutions, as well as those parts of the search tree which are symmetric to already visited states.

One way of dealing with symmetry is to add constraints that eliminate symmetric solutions (Walsh, 2012). These constraints aim to make the solutions look different even though they are the same mathematical solutions, without restricting the problem. In this section symmetry challenges in the models are identified, and corresponding symmetry breaking constraints (SBCs) are suggested. In Section 11.2 the results of testing the effect of the SBCs are presented. Symmetry breaking constraints have a number of good and bad properties (Walsh, 2012). This will be further addressed in Chapter 12.

9.1.1 General symmetry breaking constraints

In this section two symmetry issues that arise in all models are discussed, and accompanying solutions presented. The symmetry issues are related to *Route index assignment* and *Subtour elimination.*

Route index assignment. In the models, there is a lot of symmetry related to the route index assignment. The number of symmetrical solutions caused by route index assignment is given by expression (9.1), where R is the set of route indices, and A is the set of routes generated in a feasible solution. In the models the set of routes consists of the maximum potential routes that can possibly be generated. This only serves the purpose of being able to index the variables based on the routes they are in. Besides this there is no practical difference between route 1 and 2 etc, and hence our problem will have a solution space that is a lot greater that necessary.

$$\textit{\# of symmetric solutions caused by route indexing} = \frac{|R|!}{(|R|-|A|)!} \tag{9.1}$$

The number of symmetric solutions related to the route generation and assignment of ship type are equal to the possible combinations values of the binary variable y_{kr} which states whether or not ship type k sails route r, as long as the number of routes sailed by the different ship types are the same. Table 9.1 shows an example of three symmetrical solutions. The mathematical solutions are different, however they represent the same practical solution.

Below we present two different approaches of symmetry breaking constraints that help reducing the symmetry associated with the route index assignment:

Route index assignment solution approach 1 (R1):

The first solution approach (hereby referred to as *R1*) to reduce the symmetry related to route index assignment, ensures that routes are generated in ascending order and that ship types are assigned in descending order. This is done by imposing constraints (9.2). Using this approach only *Solution 3* in Table 9.1 would have provided a valid solution - this shows that the SBCs (9.2) succeed in reducing the solution space.

$$\sum_{k\in\mathcal{K}} k \cdot y_{k(r+1)} \leq \sum_{k\in\mathcal{K}} k \cdot y_{kr}, \qquad r \in \mathcal{R} \tag{9.2}$$

Constraints (9.2) reduces the symmetry related to route index assignment by stating that route r has to be created before route $r+1$ and so on. They also force routes that are sailed by ship type $k+1$ have to have lower route numbers than the routes sailed by ship type k.

$\vert K\vert = 2$		Number of routes sailed by ship type 1: $\sum_{r\in\mathcal{R}} y_{1r} = 2$					
$\vert R\vert = 10$		Number of routes sailed by ship type 2: $\sum_{r\in\mathcal{R}} y_{2r} = 1$					
Value of y_{kr}:		r**=1**	r**=2**	r**=3**	...	r**=10**	$\sum_{r\in\mathcal{R}} y_{kr}$
Symmetrical solution 1	**k=1**	1	1	0	...	0	**2**
	k=2	0	0	0	...	1	**1**
	$\sum_{k\in\mathcal{K}} y_{kr}$	1	1	0	...	1	**3**
Symmetrical solution 2	**k=1**	0	0	1	...	1	**2**
	k=2	0	1	0	...	0	**1**
	$\sum_{k\in\mathcal{K}} y_{kr}$	0	1	1	...	1	**3**
Symmetrical solution 3	**k=1**	0	1	1	...	0	**2**
	k=2	1	0	0	...	0	**1**
	$\sum_{k\in\mathcal{K}} y_{kr}$	**1**	**1**	**1**	**0**	**0**	**3**

Table 9.1: Same same, but different. The table shows three examples of symmetrical solutions related to route index assignment, i.e. the value of variable y_{kr}. Variable y_{kr} is 1 if ship type k sails route r, 0 otherwise.

Route index assignment solution approach 2 (R2):

The second approach (hereby referred to as *R2*) ensures that route r is longer or equal the length of route r+1. This way the longest routes will be assigned to the lowest route numbers and so on. This is done by constraints (9.3), which make make sure that the routes will be ordered in descending length.

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} T_{ij}\cdot x_{ijkr} \geq \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} T_{ij}\cdot x_{ijk(r+1)}, \qquad r\in\mathcal{R} \tag{9.3}$$

Comparison of the two approaches

The first approach (R1) removes most of the symmetry due to route index assignment, except from the symmetry among routes using the same ship type. If there are five routes where three routes use ship type 1 and two routes use ship type 2, the symmetrical solutions will be: $3!\cdot 2! = 12$. The second approach (R2) will erase more of the symmetry than R2. The only symmetrical solutions than can occur related to route index assignment when R2 is included is if some routes have the exact same route lengths. This is reasonable to assume that they will not have in most cases.

Subtour elimination. There is also symmetry issues related to the subtour elimination constraints. This is because in the current models an arbitrary port is set to be the depot of a non-butterfly route. This means that if a route visits five ports there are five different mathematical solutions which in reality are the same practical solution.

A way to reduce the number of possibilities is to decide that the port with the highest index is set to be the depot of a route. This reduces the symmetry of the problem because the program does not have to check if it can improve the solution by changing which port is set to be the depot.

We introduce variable p^R_{ijr}:

p^R_{ijr} - 1 if both port i and port j are visited in route r, 0 otherwise.

$$p^R_{ijr} \in \{0,1\}, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{9.4}$$

The following constraints give value to variable p^R_{ijr}:

$$r_{ir} + r_{jr} - p^R_{ijr} \leq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{9.5}$$

$$r_{ir} + r_{jr} + 2(1 - p^R_{ijr}) \geq 2, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{9.6}$$

Constraints (9.5) - (9.6) give value to variable p^R_{ijr} and ensure that it is set to 1 if both port i and port j are in route r, and 0 otherwise.

$$d_{jr} - p^R_{ijr} + 1 \geq d_{ir}, \qquad i,j \in \mathcal{P}, r \in \mathcal{R}, j > i \tag{9.7}$$

Constraints (9.7) make sure that the port within a route with the highest index is set to be the depot of the route, and can be added to the Basic and Chain model. However, in the Flower model a tiny modification is needed.

The change is needed because in a flower route we want the butterhub to be the depot, not the port with the highest index. Therefore we need to add m_{ir} to the left hand side of constraints (9.7) to relax the constraint for butterfly routes. This means that if the route is a Flower route the fact that there is a butterhub in the route must override the highest-index rule. The corresponding constraints for the Flower approach thus become constraints (9.8):

$$d_{jr} - p^{R}_{ijr} + m_{ir} + 1 \geq d_{ir}, \qquad i, j \in \mathcal{P}, r \in \mathcal{R}, j > i \tag{9.8}$$

By introducing these symmetry breaking constraints there will be only one port that is qualified to be set to the depot of a route - and hence the constraints have removed all symmetry related to the problem of arbitrarily chosen depots.

9.1.2 Flower specific symmetry breaking constraints

In this section symmetry issues and corresponding solutions in the Flower model are presented. In the Flower model there is symmetry due to the assignment of loop numbers, n_{ir}, in flower routes. This is because it is only the sequence of the loops that matter, not the numbering itself. Specifically, there will be as many symmetric solutions as there are loops in a route. This is shown in Figure 9.1.

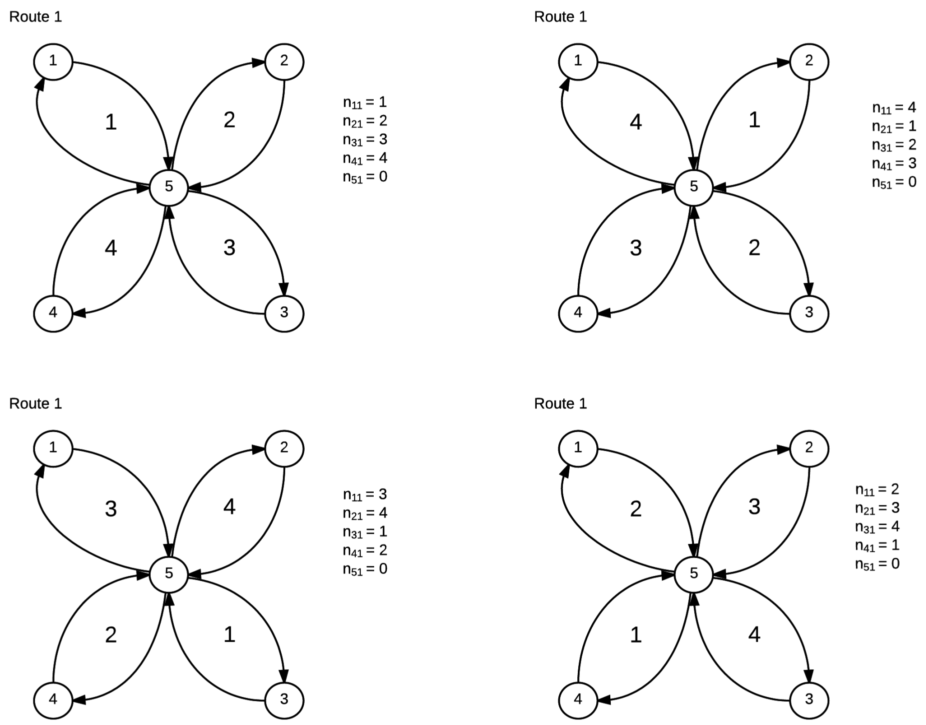


Figure 9.1: In reality the only thing that matters is the sequence of the loops, not the numbering itself. Therefore all the solutions in the figure are practically the same solution. In a Flower route with four loops, as is depicted in the figure, there will be four symmetrical solutions.

To break the symmetry related to loop numbering we want to specify the loop number of one port in the route, so that it can serve as an anchor for the rest of the ports. As long as one port has a loop number, the rest of the numbering will not have symmetric solutions, because the relative loop numbering is not random, as opposed to which loop is set to the first loop.

One way to do this is to decide on a port that is going to be in the first loop. In order to "choose" an arbitrary loop as the first loop, we need a feature that uniquely identifies a loop so that only one loop number is fixed. We chose to set the loop that contains the port with an arc from the depot, i.e. the "starting port" where $s_{ir} = 1$, with the lowest index to be in the first loop. Since no ports have the same index there will only be one loop with the lowest indexed starting port. This is illustrated in Figure 9.2.

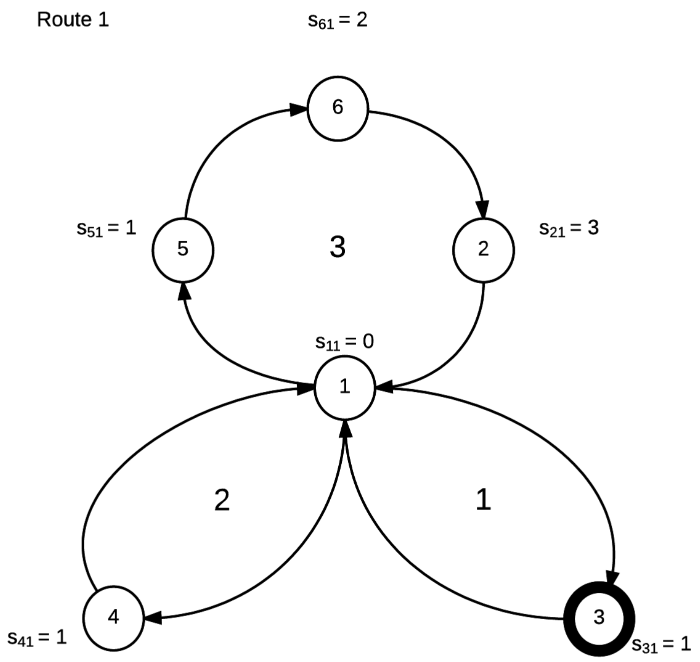


Figure 9.2: The symmetry related to loop numbering is broken by enforcing the lowest indexed starting port to be the first loop. in this case, the lowest indexed starting port is *Port 3*, and it is therefore contained in loop number 1.

Below the additions to the Flower model in order to enable the symmetry breaking constraints explained above, are given. Note that *starting port* refers to a port where s_{ir} is 1, see 9.2 for illustration.

To make the symmetry breaking constraints explained above, the additional integer variable, δ_r, and two sets of constraints, (9.9) and (9.10), are needed.

Additional variable

δ_r - equal to or less than the index of t

Additional constraints

$$\delta_r - (2 - m_{jr} - \sum_{k \in \mathcal{K}} x_{jikr}) \leq i, \qquad i, j \in \mathcal{P}, r \in \mathcal{R}, \tag{9.9}$$

$$\frac{1}{i} \cdot \delta_r - (N-1)(d_{ir} + s_{ir} - 1) - e_{ir}^F \leq 1 - \epsilon, \qquad i \in \mathcal{P}, r \in \mathcal{R}, \tag{9.10}$$

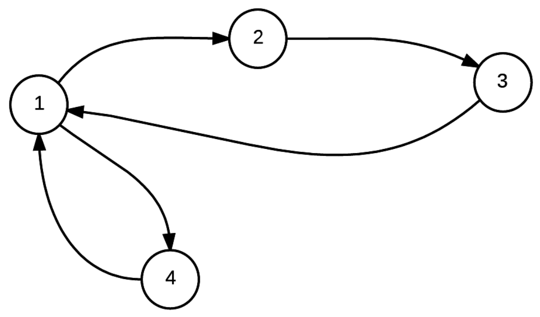
Constraints (9.9) set the value of δ_r to be less or equal to the lowest starting port index of route r. Constraints (9.10) state that if a starting port of a route, i.e. $s_{ir} = 1$, has an index in which is more than the value of δ_r, variable e^F_{ir}, which indicate whether port i is in the first loop of route r, is set to 0. Since we know that at least one loop will have to be set to be the first loop, it is not necessary with a set of constraints forcing e^F_{ir} to one because this will follow directly by the constraints presented in Chapter 6.

9.1.3 Chain specific symmetry breaking constraints

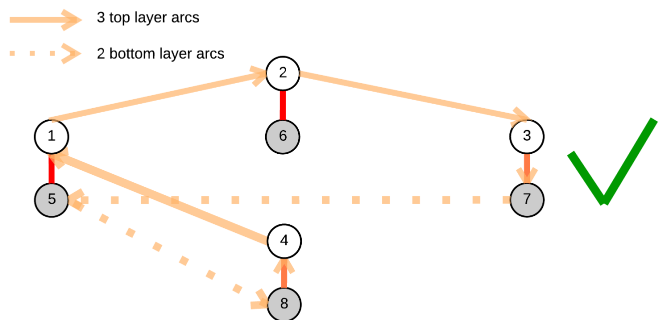
In the Chain model there are symmetry issues caused by the virtual network, i.e. the division of the ports into layers of original ports and twin ports. Since there is no logical difference between the top and bottom layer in the chain model, i.e. the original and the twin nodes, the same route can unfold itself in numerous ways in the network. We present three measures that will reduce this type of symmetry: *Top layer priority*, *One layer for simple routes* and *Restricted twin arcs usage*. In this section symmetry issues and proposed solutions in the Chain model are presented.

Top layer priority (T)

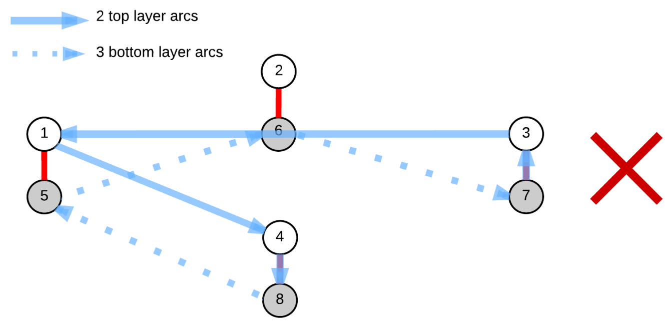
In the virtual double network all routes have equal-valued counterparts found by switching all arcs from one layer to another. This type of symmetry, illustrated in Figure 9.3, can be avoided by introducing a set of constraints that assigns a layer based on some feature of the route structure. We have chosen to assign the layer in the route with the highest number of arcs to the top layer, saying that the top layer should always be used more or as much as the bottom layer. This way constraints (9.11) ensure that there will be no ambiguity about which set of arcs should be on which layer, except from when there are exactly the same number of arcs in both layers.



(a)



(b)



(c)

Figure 9.3: The route depicted in Figure 9.3a can be realized in both ways showed in 9.3b and 9.3c. However, by imposing constraints (9.11), this symmetry is avoided by allowing only 9.3b where the number of top layer arcs are more than the number of bottom layer arcs.

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr} - \sum_{i\in\mathcal{N}\cap\bar{\mathcal{P}}}\sum_{j\in\mathcal{N}\cap\bar{\mathcal{P}}}\sum_{k\in\mathcal{K}} x_{ijkr} \geq 0, \qquad r \in \mathcal{R} \tag{9.11}$$

Constraints (9.11) state that for every route r the number of arcs in the top layer should always be more or equal to the arcs in the bottom layer. Note that the notation is the same as in Chapter 7, i.e. $\mathcal{N}$ denotes the set of both original ports and twin ports while $\mathcal{P}$ is the set of only the original ports.

One layer for simple routes (O)

A non chain-route (i.e. a simple route) has numerous of different possibilities going up and down between the two layers, not improving or altering the objective value. This symmetry challenge is illustrated in Figure 9.4. By restricting simple routes to only use one layer, this type of symmetry is avoided. Preliminary testing as well as empirical research of the shipping companies' routes indicate that the majority of the routes created will be non-butterfly routes, therefore these constraints will be activated often, and hopefully improve the solution time.

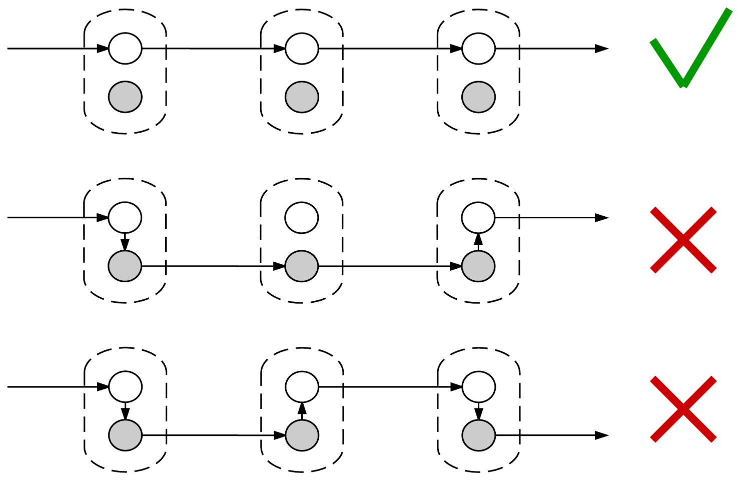


Figure 9.4: The figure shows three examples (there are many more, this is only the tip of the iceberg!) of symmetric solutions that are all valid ways of representing a simple route in the two layered network. By imposing (9.12) and (9.13) the upper solution will be preferred over the two others, i.e. simple routes will only use one layer in the network.

In order to enforce the SBCs explained above the following additional formulation needs to be included in the Chain model:

Additional variable

We introduce the binary variable, γ_{ir}, that indicates whether or not the port is visited twice or less than twice in the route. If there do not exist any variables that are visited twice in one route, the route should stay on one layer.

γ_{ir} - 1 if port i is visited twice in route r, 0 otherwise

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}(x_{ijkr}+x_{i+Nj+Nkr})-\gamma_{ir}\leq 1, \qquad i\in\mathcal{P}, r\in\mathcal{R} \tag{9.12}$$

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} (x_{ijkr} + x_{i+Nj+Nkr}) + 2 \cdot (1 - \gamma_{ir}) \geq 2, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{9.13}$$

The sets of constraints (9.12)-(9.13) give value to the new variable γ_{ir}. Constraints (9.12) force variable γ_{ir} to 1 when port i is visited twice, while constraints (9.13) set variable γ_{ir} to 0 when port i is visited less than twice in route r.

$$\sum_{i \in \mathcal{P}} \gamma_{ir} - \sum_{i \in \mathcal{P}} \sum_{k \in \mathcal{K}} (x_{i(i+N)kr} + x_{(i+N)ikr}) \geq 0, \qquad r \in \mathcal{R} \tag{9.14}$$

Constraints (9.14) ensure that that when all ports in a route are only visited once the route should stay on one layer. This is done by stating that none of the arcs between twin nodes can be used when no ports are visited twice. Note that since constraints (9.11) ensure that a route should always use the top layer more or as much as the bottom layer, the upper layer will always be used for simple routes. Thus, there is not necessary to specify which of the two layers should be used for non-chain routes.

Restricted twin arcs usage (U)

The constraints presented earlier in this section ensure that the upper layer is always used more than the lower layer, and that simple routes only use the upper layer. We also want to remove the symmetry coming from chain routes going up and down between the layers for no reason, see Figure 9.5 for illustration.

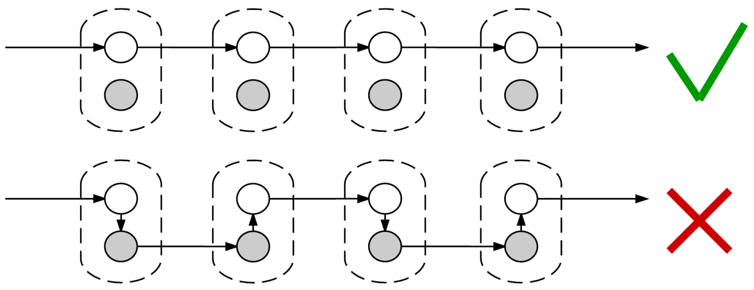


Figure 9.5: Two examples of symmetric solutions coming from a route going up and down between the layers for no reason. After constraints (9.15) are imposed only the first example will be valid.

We acknowledge that in order to create a butterhub the route needs to change layer and stay at the other layer for at least two arcs, and if the route is going up and directly up again in the neighbour node, the change of layer was of no reason. We want to enforce that a change of layer can only happen to enable a butterhub. This can be done by stating that the number of arcs a route uses in the second layer must be at least as big as the number of twin arcs it uses.

$$\sum_{i\in\mathcal{P}}\sum_{i+N\in\mathcal{N}}\sum_{k\in\mathcal{K}} x_{ijkr} - \sum_{i\in\mathcal{N}\cap\bar{\mathcal{P}}}\sum_{j\in\mathcal{N}\cap\bar{\mathcal{P}}}\sum_{k\in\mathcal{K}} x_{ijkr} \leq 0, \qquad r\in\mathcal{R} \tag{9.15}$$

Constraints (9.15) state that the number of utilized twin arcs has to be less than the number of used arcs in the lower layer.

9.2 Valid inequalities

In order to strengthen the linear relaxation of the problem, we have attempted to create valid inequalities to cut off solutions that are not integer feasible. This could make the problem easier to solve. We have created cuts that are added a priori to the formulations and also tried generating cuts dynamically as they are violated.

According to Wolsey (1998) the use of inequalities to improve formulations and obtain tighter bounds is the area in which probably the most progress has been made in the last ten years. In this section we present different constraints that can be added to the formulations in order to make them tighter. The motivation behind the valid inequalities is to get an improved optimistic bound in the root node of the B&B-tree and that this again leads to shorter computation time for larger instances of our problem. Put in another way the goal is to find effective ways to try to approximate conv(X) for the given instance. In the best case, when conv(X) is equal to the feasible area of the problem, the LP-solution will be equal to the optimal solution.

The effect of imposing valid inequalities is a trade off between a more narrow solution space and the fact that the LP-solution in the root node will take longer time to solve because of the extra constraints. Therefore, finding valid inequalities is not enough, there is a need for a computational study to figure out whether or not the valid inequalities is actually enhancing the solution time for realistic instances.

In order to find the valid inequalities that will pay off in terms of solution time we have tried to identify features of the routes that now is formulated implicitly that can be tighter with an explicit formulation.

This section is organized as follows; In Section 9.2.1 general valid inequalities that apply for all the six models are presented. In Subsection 9.2.2 and 9.2.3 model specific valid inequalities are presented for Flower and Chain models, respectively. In Section 9.3 we propose how the valid inequalities can be further strengthened by integer rounding and coefficient specification.

9.2.1 General valid inequalities

Two types of general valid inequalities, that are applicable for all models, are proposed. These are called the *Capacity miles*-valid inequality and the *Demand/capacity*-valid inequalities. In this section the formulations of the general valid inequalities are presented, in addition to an explanation of the rationale behind them.

Capacity miles valid inequality (CM)

The *Capacity miles*-valid inequality takes advantage of the known distances between the O-D pairs, and the fact that a demand at a minimum has to travel the distance between its origin port and its destination port. This is certain because the shortest path between two ports always is the direct path between them because the ports are positioned in an Euclidean 2-space. This fact can be utilized to calculate a lower bound on the total capacity of the fleet, which is exactly what is done by the Capacity miles-valid inequalities, given by constraint (9.16). The demand quantity multiplied with its direct distance makes up a lower bound on the total fleet capacity. The total fleet capacity is calculated by summarizing all vessels in use by their respective capacities and speed (in weeks). The concept behind the *Capacity miles*-valid inequality is illustrated in Figure 9.6.

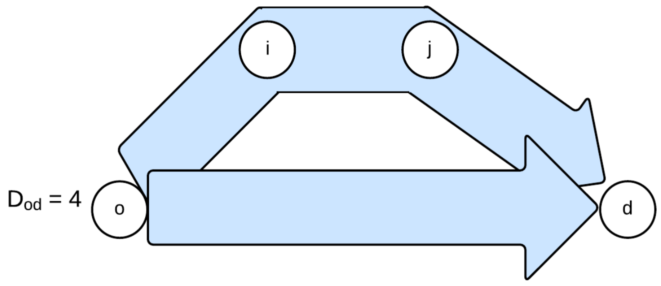


Figure 9.6: The capacity miles valid inequality exploits the fact that we know that a demand at a minimum has to travel the direct distance between its origin port and its destination port. Therefore, the direct distance between a demand multiplied with the size of the demand makes up a lower bound on the capacity.

$$\sum_{k \in \mathcal{K}} S_k \cdot V_k \cdot a_{kr} \geq \sum_{k \in \mathcal{K}} min\{S_k \cdot V_k, \sum_{o \in \mathcal{P}} \sum_{d \in \mathcal{P}} D_{od} \cdot T_{od}\} \cdot a_{kr} \geq \sum_{o \in \mathcal{P}} \sum_{d \in \mathcal{P}} D_{od} \cdot T_{od}, \tag{9.16}$$

The formulation of the *Capacity miles*-valid inequality is given by onstraint (9.16). A stronger versions of the constraint is obtained by replacing the product of ship type dependent parameters capacity and velocity, i.e. $S_k \cdot V_k$, with the minimum of this and the total demand in the system multiplies with its respective direct distances. The reason why this is possible is that there will *never* be necessary to use the exceeding capacity of a vessel if this is greater than the total demand in the whole system.

Demand/capacity valid inequalities (D/C)

The *Demand/capacity (D/C)*-valid inequalities exploit the fact that if the ports are split into two subsets we know that the flow between the subsets at least has to equal the total demand between them. This knowledge can be used to determine a lower bound on the capacity of the ships sailing between subset 1 and subset 2 - and thus also a lower bound on the arcs needed to be used between the two subsets. The concept is illustrated in Figure 9.7a. In the following two versions of the D/C-valid inequalities will be presented: the *Simple D/C*-valid inequalities and the *D/C flow-valid inequalities.*

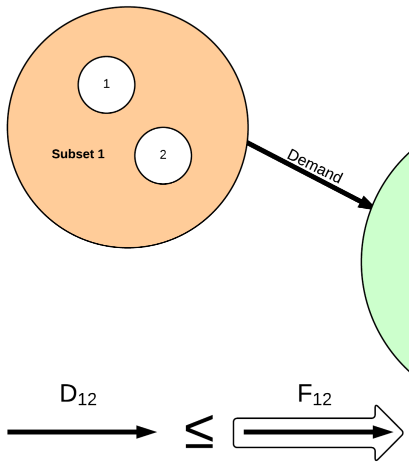
Simple D/C valid inequalities

$$\sum_{i\in\mathcal{P}^{\mathcal{A}}}\sum_{j\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} S_k\cdot x_{ijkr} \geq \sum_{i\in\mathcal{P}^{\mathcal{A}}}\sum_{j\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} min\{S_k, \sum_{i\in\mathcal{P}^{\mathcal{A}}}\sum_{j\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}} D_{ij}\}\cdot x_{ijkr} \geq \sum_{i\in\mathcal{P}^{\mathcal{A}}}\sum_{j\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}} D_{ij}, \qquad \mathcal{P}^{\mathcal{A}}\subset\mathcal{P} \tag{9.17}$$

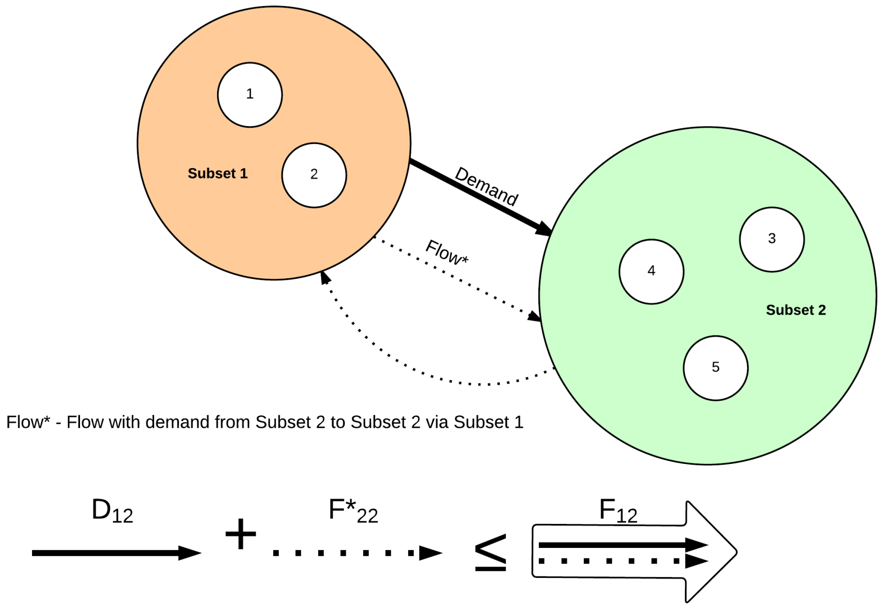
The simple D/C-valid inequalities are given by constraints (9.17). The understanding of these constraints are that the set of ports is divided into two non-overlapping and extensive subsets non of them being the empty space and that the flow between the subsets has to be greater or equal to the net demand between them. As for the CM-valid inequality, a stronger versions of the constraint is obtained by tightening the parameters. In this case this is done by replacing the vessel capacity parameter, S_k, with the minimum of this and the total demand between the two sets, $\sum\limits_{o\in\mathcal{P}}\sum\limits_{d\in\mathcal{P}} D_{od}$. This can be done because if the total demand between the two subsets is lower than some of the vessel capacities, the exceeding capacity is in fact irrelevant, and using the demand instead makes a tighter bound on the constraint.

D/C flow-valid inequalities (D/C-f)

The second version of the D/C-valid inequalities is based on the same concept as the simple version, but they are further strengthened by exploiting that the flow going via Subset 1 to Subset 2 can be added on RHS of the constraints. This is because if there is external flow coming in to subset 1 with final destination subset 2 this may have implications for the lower bound of the vessel capacity between the two subsets, because this demand also needs vessel



(a) Simple D/C-valid inequalities



Subset 1
1
2
Demand
Flow*
Subset 2
3
4
5
Flow* - Flow with demand from Subset 2 to Subset 2 via Subset 1
D12
+
F*22
≤
F12

(b) D/C flow-valid inequalities

Figure 9.7: The D/C valid inequalities exploit the fact that the total flow from Subset 1 to Subset 2 must be at least equal to the net demand from Subset 1 to Subset 2, and vice versa. By calculating this we get an immediate lower bound for the total capacity of the fleet between the two subsets of ports. The *D/C flow-valid inequalities* extends this concept further by stating that the total flow from Subset 1 to Subset 2 must be at least equal to the demand from Subset 1 to Subset 2 and the flow going to Subset 2 via Subset 1, with destination in Subset 2.

capacity to return to its correct subset. The *D/C-f*-valid inequalities are given by constraints (9.18). Note that for the D/C-flow-valid inequalities the same parameter substitution ($S_k \implies \sum\limits_{o \in \mathcal{P}} \sum\limits_{d \in \mathcal{P}} D_{od}$) as was done for the Simple D/C is not possible. This is because it is not possible to know an upper bound on the flow from Subset 1 to Subset 2, because in theory a demand can cross the line between two subsets an infinite number of times.

$$\sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} S_k \cdot x_{ijkr} \geq \sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} D_{ij} + f_{jiodkr}, \qquad \mathcal{P}^{\mathcal{A}} \subset \mathcal{P} \tag{9.18}$$

Formulation of the D/C valid inequalities in Chain model

In the Chain model some minor notational adjustments are needed in the formulations of the D/C-valid inequalities. the Simple D/C-valid inequalities are given by constraints (9.19) instead of constraints (9.17), and the D/C valid inequalities with flow are given by constraints (9.20) instead of (9.18). The only difference is that the flow variable x_{ijkr} is replaced with $x_{ijkr} + x_{i+Nj+Nkr}$ because the flow in both layers needs to be included.

$$\sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} \sum_{k \in \mathcal{K}} min\{S_k, \sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} D_{ij}\} \cdot (x_{ijkr} + x_{(i+N)(j+N)kr}) \geq \sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} D_{ij}, \quad \mathcal{P}^{\mathcal{A}} \subset \mathcal{P} \tag{9.19}$$

$$\sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} S_k \cdot (x_{ijkr} + x_{(i+N)(j+N)kr}) \geq \sum_{i \in \mathcal{P}^{\mathcal{A}}} \sum_{j \in \mathcal{P} \cap \overline{\mathcal{P}^{\mathcal{A}}}} D_{ij} + f_{jiodkr}, \qquad \mathcal{P}^{\mathcal{A}} \subset \mathcal{P} \tag{9.20}$$

9.2.2 Flower specific valid inequalities

In this section we present valid inequalities that can be added to the Flower and Flower TT models to make the formulations stronger. As for the symmetry breaking constraints, the problem specific valid inequalities in the Flower model are also related to the loop numbering in a flower route.

Only one first/last loop-valid inequalities (L)

We know that in a Flower-route, there will never be more than one loop that is set to be the first loop or last loop, i.e. where the loop number, n_{ir}, has the value 1 or l_r, respectively. This is not explicitly stated in the model, so specifying this as a valid inequalities might be beneficial for the run time.

The *Only one first/last loop*-valid inequalities are state that the variables indicating if port i is in route r's first or last loop, e^F_{ir} and e^L_{ir}, can only take the value 1 for one of the "starting ports", i.e. a port i having sequence number $s_{ir} = 1$. Even though the concept of the *Only first/last loop*-inequalities is almost banal, the formulation is not straightforward, and a bundle of new variables and constraints are required:

Additional variables

To construct the *Only one first/last loop*-valid inequalities two extra binary variables are required, κ_{ir} and μ_{ir}:

κ_{ir} - 1 if port i is a starting port in route r, i.e. if s_{ir} =1, 0 otherwise

μ_{ir} - 1 if port i is a starting port in the first or/and last loop of route r, 0 otherwise

Variable κ_{ir} takes on the value 1 when port i is a starting port in route r, i.e. when $s_{ir} = 1$, and 0 otherwise. Variable μ_{ir} is 1 when port i is a "starting port", i.e. $\kappa_{ir} = 1$, *and* port i is in the first or/and last loop of route r, i.e. $e^F_{ir} = 1$. For simple routes the loop is both the first and last loop of the route.

Additional constraints

$$s_{ir} + \kappa_{ir} + 2d_{ir} + 2(1 - \sum_{k \in \mathcal{K}} y_{kr}) + 2r_{ir} \geq 2, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{9.21}$$

$$s_{ir} + (N-1)\kappa_{ir} \leq N, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{9.22}$$

Constraints (9.21)-(9.22) give value to κ_{ir}. Constraints (9.21) make sure that κ_{ir} is set to 1 when port i is a starting port, i.e. $s_{ir} = 1$. In the constraints, κ_{ir} is forced to be 1 if $s_{ir} \leq 1$, thus we have to relax the constraints for the cases where the node is not in the route ($r_{ir} = 0$), the node is a depot ($d_{ir} = 1$) or when the route does not exist ($y_{kr} = 0$). Constraints (9.22) force it to 0 when the sequence number $s_{ir} > 1$.

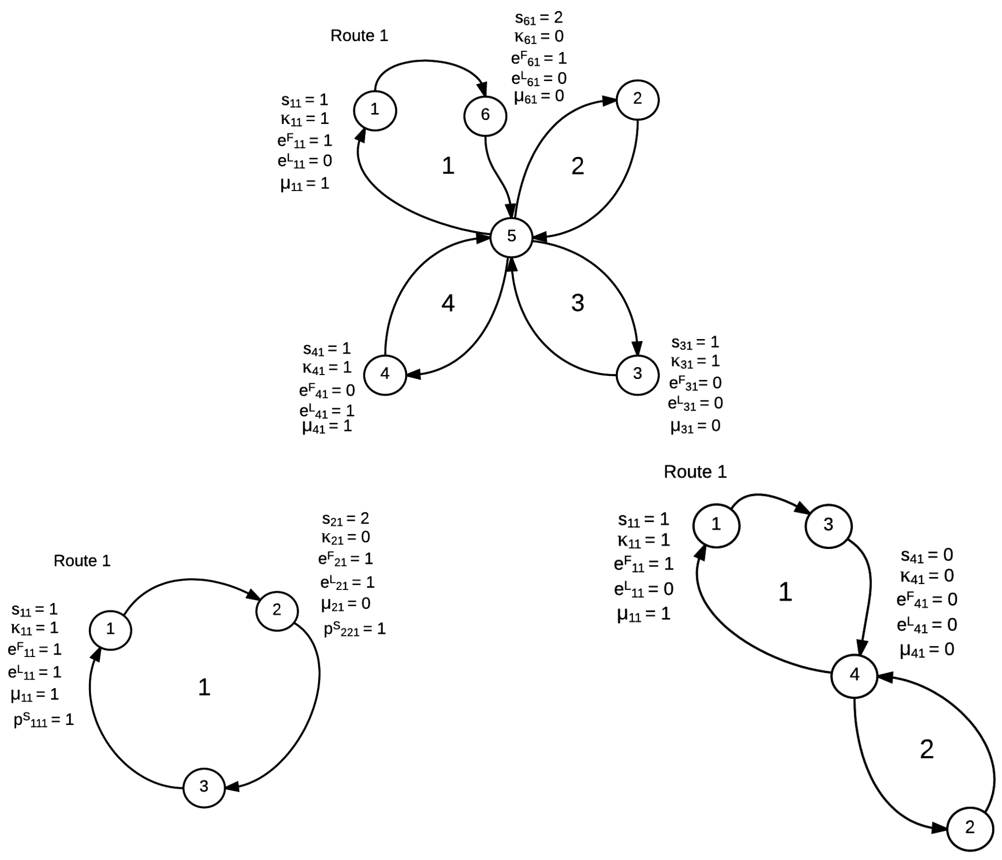


Figure 9.8: Illustration of how the μ_{ir}-variable and the κ_{ir}-variable are set for different port/route-combinations. κ_{ir}-variable is 1 when port i is a starting port, and μ_{ir} is 1 when port i is the starting port in the first and/or last loop of route r, for all other ports or routes μ_{ir} is 0.

$$\kappa_{ir} + e^F_{ir} + e^L_{ir} - p^S_{iir} - \mu_{ir} \leq 1, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{9.23}$$

$$\sum_{i \in \mathcal{P}} \mu_{ir} + 2(1 - \sum_{k \in \mathcal{K}} y_{kr}) = 2, \qquad r \in \mathcal{R} \tag{9.24}$$

Constraints (9.23)-(9.24) give value to variable μ_{ir}. Constraints (9.23) ensure that μ_{ir} gets the value 1 if both $\kappa_{ir} = 1$ and $e^F_{ir} = 1$ and/or $e^L_{ir} = 1$, i.e. when port i is the starting port in route r's first and/or last loop. If the route is a simple route with only one loop, e^F_{ir} and e^L_{ir} will both be 1, thus we have to subtract 1 when this is the case, so the constraints hold. That is the reason for subtracting p^S_{iir}, which is 1 when port i is in both the first loop and the last loop at once. Constraints (9.24) ensure that exactly two μ_{ir} per route can have the value 1, in other words, only one starting port in each route can be in the first loop, and only one starting port can be in the last loop. However, one starting port can be in both the first and last loop. This happens for simple routes because they have only one loop.

9.2.3 Valid inequalities for the Chain model

In this section we present valid inequalities that can be added to the formulations of the Chain and Chain TT models to make them stronger. As for the symmetry breaking constraints, the valid inequalities for the Chain models are also related to the two layered virtual network.

Two single visits (V)

In a chain route there must be at least two ports that are only visited once, as Figure 9.9 shows. This comes from layer structure that is used to enable chain routes. Since this is a known feature the model might as well get this explicitly through a valid inequality.

$$\sum_{i \in \mathcal{P}} \gamma_{ir} \leq \sum_{i \in \mathcal{P}} r_{ir} - 2, \qquad r \in \mathcal{R} \tag{9.25}$$

The set of constraints (9.25) states that in the Chain model at least two ports must be visited only once. It uses variable γ_{ir} introduced in Section 9.1.3 that is 1 if port i is visited twice in route r.

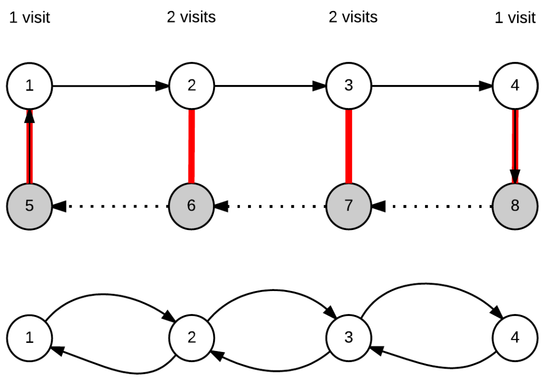


Figure 9.9: The figure shows that in the Chain model every route must have at least two ports that are visited only once. This is because that in complex routes both layers are needed, and there have to be two "transportation ports" that the route use to move up and down between the layers, and thus these two ports cannot be visited more than once.

9.3 Integer rounding valid inequalities

In this section no new problem specific valid inequality concepts are introduced, however, we show how some of the valid inequalities presented in Section 9.2.1-9.2.3 , can be further strengthened with applying integer rounding.

Integer rounding inequalities arise in the context of (mixed-)integer programs, that is to say, optimization problems where some of the variables are required to take integral values. They exploit the knowledge about some variables being integer to round up both sides of a constraint, hopefully producing a tighter bound. This is sort of a way of telling the LP-relaxation of the problem that this variable is in fact integer. The goal of the MIR is to round up one side of the inequality as much as possible, and the other side as little as possible, creating a tighter bound. This is illustrated in Figure 9.10. In order to obtain an inequality that is suitable for integer rounding the whole constraint is divided by a number that ensure that the right hand side is rounded up as little or as much as possible, depending on the constraint being a $\geq$- or $\leq$-constraint, respectively.

Procedure. The general procedure of deriving a MIR constraint is as follows; we are given a constraint of the form:

$$x + Cy \geq b, \quad \text{where} \quad y \text{ is integral}, \quad x \geq 0, \quad C > 0 \text{ and } b \text{ are given values} \tag{9.26}$$

To derive a MIR inequality we calculate a help parameter α:

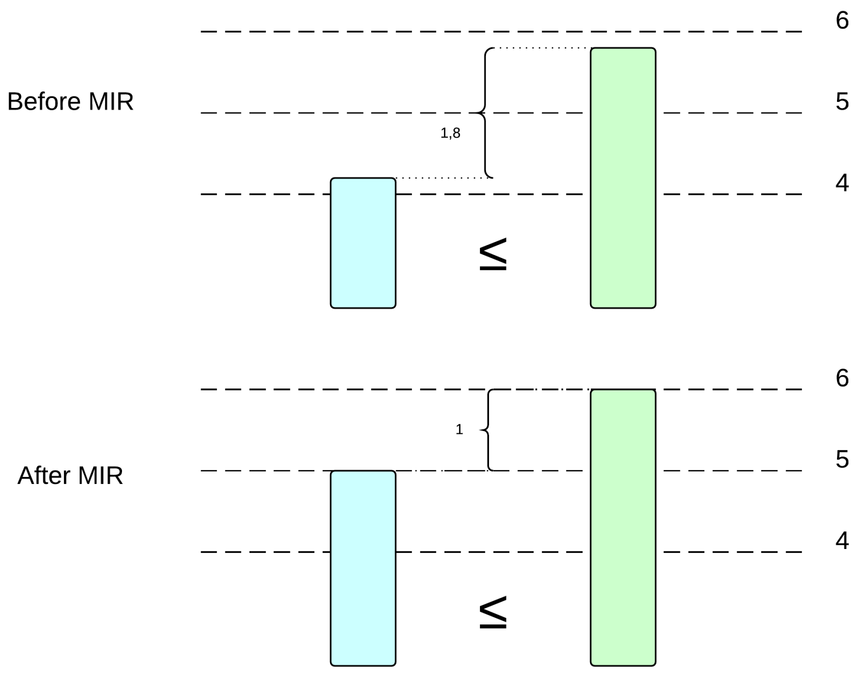


Figure 9.10: The goal with integer rounding is to round up one side as much as possible, and the other side as little as possible and this way obtain tighter bound.

$$\beta + \epsilon = \lceil \frac{b}{\alpha} \rceil, \qquad \beta \quad \text{is integral} \tag{9.27}$$

The MIR inequality is obtained by dividing (9.26) by the α gotten from (9.27):

$$\lceil \frac{x + Cy}{\alpha} \rceil \geq \lceil \frac{b}{\alpha} \rceil \tag{9.28}$$

The original constraint, (9.26), has two variables, x and y, where x is nonnegative and y is integer. To obtain the MIR we need a parameter, α, that is such that the right handside becomes 1.001 (or another number that would be rounded up with almost 1 by the ceiling function). By dividing both sides (9.26) by this α-parameters found from (9.27) we obtain the mixed inequalities in 9.28. Because of the way the α is calculated in (9.27) we can be sure that the right hand side of the constraint will be rounded up a lot ($\approx$ 1) by the ceiling function. Hopefully, and most probably, the left hand side will be rounded up by less, all this leading to a tighter constraint.

Different β-values give different α-values that give different integer rounding valid inequalities. We want the α-parameter to be such that the rounding valid inequalities get the most effect. This happens when the right side of the constraint is just above some integer value forcing the left side to round up by almost 1.

Below the Capacity miles- and Demand/Capacity-valid inequalities are presented:

CM-valid inequality with integer rounding. Constraint (9.29) gives the Capacity miles-valid inequality strengthened with integer rounding.

$$\sum_{k\in\mathcal{K}} \left\lceil \frac{min\{S_k, \sum_{o\in\mathcal{P}^{\mathcal{A}}} \sum_{d\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}\}} \cdot V_k}{\alpha} \right\rceil \cdot (\sum_{r\in\mathcal{R}} a_{kr}) \geq \left\lceil \frac{\sum_{o\in\mathcal{P}} \sum_{d\in\mathcal{P}} D_{od}\cdot T_{od}}{\alpha} \right\rceil \tag{9.29}$$

Simple D/C valid inequalities with integer rounding. Constraints (9.30) express the simple D/C-valid inequalities with integer rounding.

$$\sum_{i\in\mathcal{P}^{\mathcal{A}}} \sum_{j\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} \left\lceil \frac{min\{S_k, \sum_{o\in\mathcal{P}^{\mathcal{A}}} \sum_{d\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}} D_{od}\}}{\alpha} \right\rceil \cdot x_{ijkr} \geq \sum_{i\in\mathcal{P}^{\mathcal{A}}} \sum_{j\in\mathcal{P}\cap\overline{\mathcal{P}^{\mathcal{A}}}} \left\lceil \frac{D_{ij}}{\alpha} \right\rceil, \quad \mathcal{P}^{\mathcal{A}} \subset \mathcal{P} \tag{9.30}$$

Chapter 10

Instance generation

In this chapter the background, construction and reference scheme for the test data used for the computational study in Chapter 12 are presented.

The aim of the computational study is twofold; one is to explore how well the models perform in terms of scalability and running time, and the second is to evaluate the sensibility of the solutions they make. The intention of the test instances is thus to facilitate a meaningful and conducive analysis in the computational study, so that it gives a real impression of how they work and what are their main improvement areas. In order to learn as much as possible about the models it is therefore key to have test instances that differ in structure and size.

The Basic, Flower and Chain models use the same format for the input data, while the models with transit time extensions need separate input files.

The instances are comparable in size to the instances used in recent literature that presents exact models for solving the LS-NDP, i.e. (Reinhardt and Pisinger, 2011). The number of ports range from 4 to 15, while the number of ship types is either 2 or 3 and the number of demands range from 5 to 12.

First, in Section 10.1, the reference scheme for the data instances is presented. In Section 10.2 an overview of the parameters related to the different ship types and ports is given. In Section 10.3 there is a description of the three instance types in terms of port positions and the distribution of demands; namely the General, Feeder and Hub and Spoke. This section also contains an explanation of how the instances are generated. In the last section, Section 10.4, a specification of the instances used for the Transit time-models is given.

10.1 Instance reference scheme

The test instances vary in four dimensions:

1. *The number of ports*
2. *The number of ship types*
3. *The number of demands*
4. *The network type*

The domains of the four dimensions and the instance reference scheme are shown in Table 10.1. The three former dimensions (number of ports, ship types and demands) are quite self explanatory, while the latter dimension, *network type*, on the other hand needs some further explanation. This is given in Section 10.3. Table 10.1 shows the domain of the four dimensions specifying a test instance. It also shows the reference scheme used to uniquely refer to the specific instance for the original and transit time models. Example; a test instance for an original model with a Feeder-structure, five ports, two ship types and seven demands will be denoted F[527]. A test instance for a transit time model with Hub and spoke-structure with seven ports, three ship types and eight demands is denoted T-H[738]

Instance reference scheme		
Data instance original models:		Type[#Ports, #Ship types, #Demands]
Data instance transit time models:		T-Type[#Ports, #Ship types, #Demands]
Where:	Network type	∈ { **F** (Feeder), **H** (Hub and spoke),**G** (General)}
	# Ports	∈ [4,15]
	# Ship types	∈ [2,3]
	# Demands	∈ [4,12]

Table 10.1: Instance reference scheme and domains of the four dimensions specifying a particular test instance.

10.2 Parameters

In this section the parameters used in the computational study are presented.

The parameters are not based on real data. The reason for this is that considering the models' high degree of simplification it would not be meaningful to give any real interpretation of the numbers. For instance there would be no point in exploring how much Maersk would gain from applying these models compared to the services they offer today. However, the relative difference between the parameters are meant to encourage and facilitate different solutions for different settings. For example; the cost of using ship type # 3 is higher than using ship type # 1, however if it is fully loaded the cost per container will be lower than by using ship type # 1. This way it is possible to evaluate to which degree the models make rational decisions and work the way they are intended.

As was explained in Section 10.1 an instance is identified by type, number of ports, number of ship types and number of demands. If an instance has 5 ports these ports are port # 1, port #2, $\cdots$, port # 5, and if it has 6 ports these are port # 1, port #2, $\cdots$, port # 6, and so on. The same goes for ship types. Table 10.2 shows the *Capacity*, *Speed* and *Sailing cost* data of each ship type.

Ship type	Capacity	Speed	Sailing cost	Number of vessels
1	5	5	10	20
2	10	9	16	15
3	22	8	18	12

Table 10.2: Ship type parameter used in test instances. Since the data is not based on real numbers the units are not specified.

Port number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Transhipment cost	1	3	1	1	6	5	2	3	1	2	1	2	1	5	1

Table 10.3: The table shows the transhipment costs per container shipped for the 15 ports used in the test instances. The transhipment cost does not depend on the ship type, it is only dependent on the port.

Visiting costs	*Ship type 1*	*Ship type 2*	*Ship type 3*
Port 1	1	1	1
Port 2	1	3	4
Port 3	2	2	3
Port 4	1	1	2
Port 5	1	1	1
Port 6	3	4	5
Port 7	1	2	3
Port 8	1	2	2
Port 9	2	2	2
Port 10	4	5	7
Port 11	1	2	2
Port 12	2	2	3
Port 13	1	2	2
Port 14	2	3	3
Port 15	1	2	4

Table 10.4: Visiting costs for each combination of the 15 ports and the three ship types. In general Ship type 1 has the lowest visiting costs and Ship type 3 the most expensive visiting costs. This has to do with the size of the vessels, hence the capacity that can be seen in Table 10.2.

10.3 Network types

The optimal network of routes depend on the geographical spread of the ports as well as how the demand is distributed among them. Since we aim to make our models as general as possible, we tested them on three types of data sets with different structures that often appear in liner shipping and possess different characteristics. As seen in Table 10.1 *network type* is one of the characteristics used to describe a test instance.

The interpretation of a network type is the structure when it comes to the position of the ports and the distribution of the demand between them. The three network types used are called the Feeder network, the the Hub & Spoke-network and the General network. The networks are illustrated in Figures 10.1a-10.1e.

The Feeder-network

A feeder line is a peripheral route or branch in a network, which connects smaller or more remote nodes with a route or branch carrying heavier traffic. Figure 10.1a shows an example of a feeder-network. An example can be the airline traffic in Norway where a substantial part of the airline traffic is between Oslo and smaller cities. Since it would not be economical to have a direct route between Oslo and all other cities it often happens that the route stops at 2 or 3 smaller places before returning to Oslo.

An example of this is Widerøe's route going Oslo-Ørsta Volda-Sandane-Oslo. The advantages with this is better exploitation of the plane capacity and lower fuel costs (and other variables costs caused by a reduced flight distance). The disadvantages is that the passengers going from Oslo to Sandane or from Ørsta Volda to Oslo have to make a transfer and thus the transit time becomes higher. How the feeder-instances are generated is illustrated and explained in Figure 10.1b.

The Hub & Spoke-network

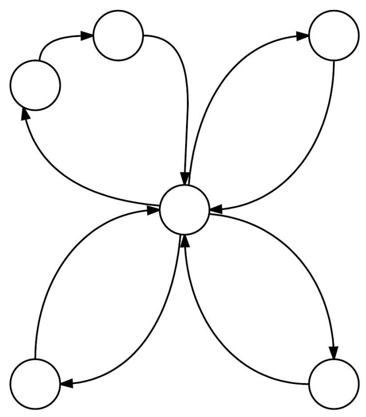
The Hub and Spoke-network can be viewed as a network of several Feeder-networks, see Figure 10.1c. It is characterized by a network with several clusters of nodes where each cluster has a hub. A Hub and Spoke-network is often effective when there exist collections of ports that are geographically separated, and there is demand internally in the clusters as well as between them.

An example of this is the global shipping network shipping goods between the continents. Typically large vessels sail between the hubs of the continents (i.e. Shanghai and Rotterdam), while smaller vessels sail inside one continent. How the Hub and Spoke-instances are generated is illustrated in Figure 10.1d. In the H&S-instances there are two clusters of ports that are placed in a circle with radius 5. Half of the ports are in one cluster and the rest in the other.

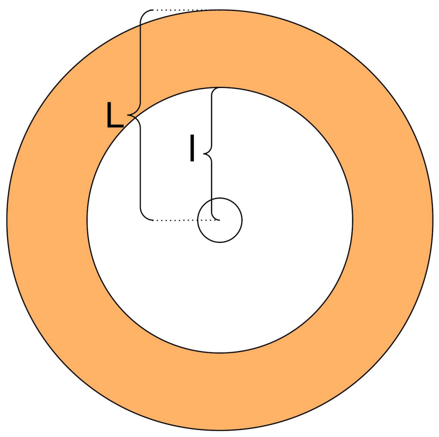
The distance between the two clusters are 20. This means that inside of a cluster the maximum distance between two ports is 10, while for ports in different clusters the maximum distance is 40.

The General-network

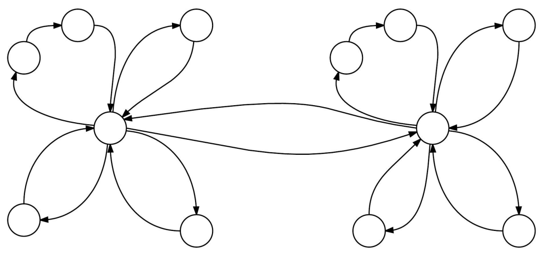
The general liner shipping networks do not have a specific structure, and the sets used in the testing contain randomly distributed demands and positions. Figure 10.1e is an example of a route in a general liner shipping network. We model this by placing all ports randomly within a squared area within a length range [0,L] and a breadth range [0,L]. This is described in Figure 10.1f.



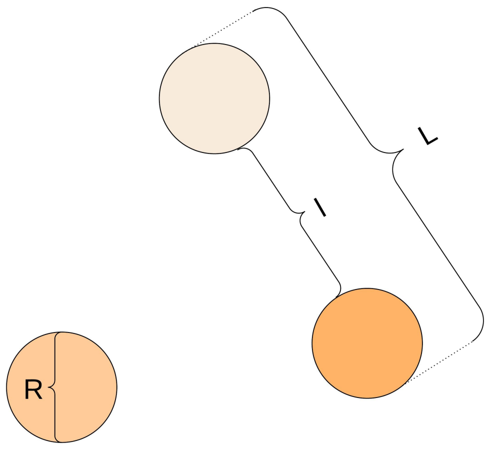
(a) Feeder network



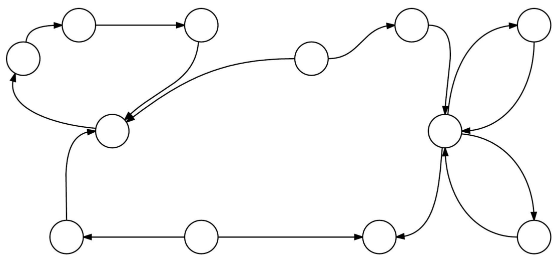
(b) Generation of Feeder instance



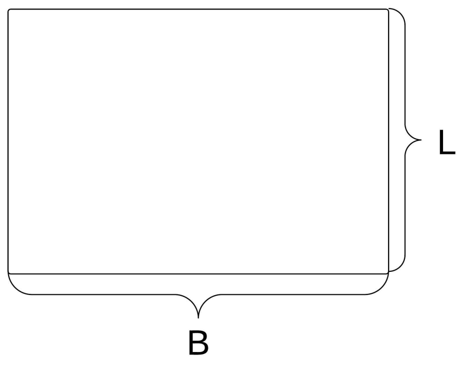
(c) Hub and Spoke-network



(d) Generation of Hub and Spoke-instance



(e) General network



(f) Generation of general instance

Figure 10.1: Hub and Spoke-instance. Figure 10.1e shows a typical General network. The General network is characterized of the ports being spread all over with no clear clusters of ports. Figure 10.1f illustrates the generation of a General instance. he general instance is generated by placing the ports randomly in BxL squared area. In our cases both B and L are set to 10. The demand is randomly distributed with random values ranging from 1 to 9.

10.4 Transit time instances

As seen from Table 10.1 a transit time-instance is referenced by T-*Type*[# of ports, # of ship types, # of demands]. The data instances for the transit time models are almost similar to the ones for the general models, however there are some exceptions.

First of all; the transit time instances contains data about the maximum allowed transit time for each order. A drawback with the functionality of the transit time models compared to the original models is that the flow variable, $f_{ijod\omega kr}$ is binary rather than integer. However, if each container was seen as its own order this would be equivalent to in the original models. In each transit time instance the orders are bundled in sets of one, two or nine containers - leading to different degrees of "flowability". We denote the size of the bundle the *order split size* (OSS). An other aspect to notice is that in the transit time models the notion of *a demand* is still all demand from one port to another port, independently of how many orders each demand is divided into.

In the computational study of Transit Time models it is interesting to see how the models behave when some orders have low transit time requirements and other orders do not. The feasibility of the transit time requirements are ensured by setting it as the time ship type # 1 spends on sailing the direct distance of from order ω's origin port o to its destination port d. We also want to investigate how the general transit time requirements influence the solution and the run time of a particular instance. In order to facilitate the investigation of these aspects there are three versions of each instance; *Urgent*, *Moderate* and *Uncritical* dependent of the transit time requirements on the orders. In the Moderate version of an instance all the transit times are considerably higher than the transit time requirements of the Urgent instance. In the *Uncritical* version of an instance the transit times are very high, and will very rarely have any effect. How the transit times of the different versions of the instances are set is shown in Table 10.5.

Transit time urgency levels	Group 1 (1/3 of all orders)	Group 2 (1/3 of all orders)	Group 3 (1/3 of all orders)
Urgent	0.7	0.8	1
Moderate	1	1.2	1.4
Uncritical	2.8	3.6	6

Table 10.5: Transit time urgency levels for different versions of a transit time instance. The numbers in the table represent the transit time requirement as this number multiplied with the sailing time ship type # 1 spends on sailing the *direct* distance between the origin and the destination of the particular order. For each urgent level the orders are divided into three groups with different transit time requirements (Group 1 has the lowest transit time requirement and Group 3 has the largest).

Chapter 11

Evaluation of strengthening formulations

Before the main computational study was carried out, we performed a thorough preliminary testing of the suggested measures of strengthening the formulations presented in Chapter 9. The purpose of the testing was to explore the effect of the symmetry breaking constraints (SBCs) and valid inequalities, and to use this to conclude on which of the strengthening formulations that should be included in the different models. Table 11.1 gives an overview of the SBCs and valid inequalities that were tested.

Testing approach. Ideally, every single combination of SBCs and valid inequalities should have been tested on all the six models on all instances. Since this is not practically feasible, we have attempted to perform the testing in a logical order and with test cases that would give the best basis for decisions on the performance of the strengthening formulations. Detailed descriptions of how the symmetry and valid inequality-testings were performed, are given in their respective sections. Anyhow, on a high level: First, the symmetry elimination constraints (SBCs) were tested. The SBCs that had good performance were added to the models for further evaluation of the valid inequalities. The strengthening formulations have been tested on the original models, and the general SBCs and valid inequalities have been tested on the Basic model. All the test instances in this section have been of type General.

A note on evaluation principles. There is no recipe for how one can evaluate the effect of strengthening formulation, and therefore we would like to convey some of the principles that pervade the assessment of the strengthening formulations in this thesis. Our overarching motivation for adding the strengthening formulations is to reduce the actual run time of the models. It is not to show some theoretical feature of the valid inequalities. We are interested in the practical gains from the valid inequalities, not the theoretical. If the valid inequalities perform well when Xpress presolve is disabled, but does not perform well when it is enabled, we have not included it in the model.

Even though it is the run times that we really want to reduce, looking only at run times is not enough. We are looking for the SBCs and the valid inequalities that will enhance the run times of large instances. For small instances, adding strengthening formulations will seldom enhance solution time. In the evaluation of the SBCs and valid inequalities we have therefore also taken the number of nodes and LP objective values into account. We have especially emphasized how the strengthening formulations scale, and if they show a relatively positive trend as the instances get larger.

Outline. This chapter is structured as follows; In Section 11.1 the model reference scheme is presented, making it possible to unambiguously refer to a certain model containing a specific combination of symmetry breaking constraints and valid inequalities. The results from the testing of the SBCs and valid inequalities are presented in Section 11.2 and Section 11.3, respectively.

11.1 Model variant reference scheme

The numerous SBCs and valid inequalities that have been tested in the computational studies can easily be a source of confusion discussing the computational results. Therefore, there is a need for an explicit way of referring to a certain model containing a specific combination of strengthening formulations. Table 11.1 shows the model reference scheme with the different codes used to denote the symmetry breaking constraints and valid inequalities. This way there will be no doubt about which results that are discussed and analyzed. As an example of the use of the reference scheme: Take an original Flower model containing the route index assignment-SBC that sort the routes by length (R2) and the Flower specific valid inequality *Only one first/last loop* (L), will be referred to as: F[R2,L].

MODEL VARIANT REFERENCE SCHEME

Where:	Model reference:	Model code[SBCs code, Valid inequalities code]
	Model code:	∈ {Basic(**B**), Flower (**F**), Chain (**C**), Basic TT (**BT**), Flower TT (**FT**), Chain TT (**CT**)}

Strengthened formulations			Code	Constraints
SBCs				
	General	Route index assignment		
		- Approach 1	**R1**	(9.2)
		- Approach 2	**R2**	(9.3)
		Subtour elimination	**S**	(9.7) (Basic and Chain), (9.8) (Flower)
	Flower specific	Loop numbering	**L**	(9.9)-(9.10)
	Chain specific	Top layer priority	**T**	(9.11)
		One layer for simple routes	**O**	(9.14)
		Restricted twin arcs usage	**U**	(9.15)
Valid inequalities				
	General	Capacity miles with integer rounding (CM)	**C**	(9.29)
		Simple Demand/Capacity (D/C)	**D**	(9.17) (Basic and Flower), (9.19) (Chain)
		D/C with flow (D/C-f)	**F**	(9.18) (Basic and Flower), (9.20) (Chain)
		D/C with integer rounding (D/C-IR)	**IR**	(9.30)
		Xpress preprocessing	**X**	
	Flower specific	Only one first/last loop	**N**	(9.21)-(9.24)
	Chain specific	Two single visits	**V**	(9.25)

Table 11.1: The table shows the reference scheme used to uniquely refer to a model with specific SBCs and valid inequalities. Overview of the SBCs and the valid inequalities that are evaluated in the computational study, and how they are referred to in the model reference scheme.

11.2 Symmetry testing

In this section the results from the testing of the symmetry breaking constraints (SBCs), presented in Section 9.1, are described, and the effectiveness of each SBC is evaluated. The final outcome of the testing, i.e. which SBCs that were added to the models, is shown in Table 11.2.

SBCs		**Outcome**
General:	R1	✗
	R2	✓
	S	✓
Flower specific:	L	✓
Chain specific:	T	✓
	O	✓
	U	✗

Table 11.2: Outcome of symmetry testing

Testing of the general symmetry breaking constraints

The testing of the general SBCs presented in Section 9.1.1 are done with the Basic model. This is because all the other five models are based on this model, and we think that it is likely that the conclusions from the testing on the Basic model also would apply for these. In addition, since the Basic model is the least complex of the models the conclusions of symmetry testing can come from smaller instances than for the other models. This is because the other models have more variables, and therefore the gains from the SBCs might come for larger instances.

Two general symmetry issues are discussed in Section 9.1; symmetry related to route index assignment and subtour elimination. For the route index assignment-symmetry two approaches were proposed; $R1$ given by constraints (9.2) and $R2$ given by constraints (9.3). For the symmetry in subtour elimination there was one, S given by constraints (9.7). The symmetry testing was done by running the Basic model with all possible combinations of the SBCs on 9 instances of varying sizes. $R1$ and $R2$ are mutually exclusive, because they are both measures to reduce the symmetry due to the route indexing. Because of this, the symmetry testing consisted of testing the three SBCs separately, as well as the combinations $R1 + S$ and $R2 + S$.

Table B.1 shows the run time the different combinations of SBCs gave for different instances. The symmetry testing shows that including the symmetry breaking constraints is effective and help reduce the run time. The results indicates that the best combination of SBCs is *R2 + S*. The symmetry testing were performed on instances with two ship types, from 5-9 ports and 5-9 demands. The instances were categorized into *Small* (G[525], G[527], G[725]), *Medium* (G[529], G[727], G[729]) and *Large*(G[925], G[927], G[925]). For each instance the SBC combinations were ranked from 1 (fastest) to 6 (slowest) based on the run time on that particular instance. Table 11.3 shows the average ranking for each SBC combination on the Small, Medium and

Average ranking:	**Run time**			**# of nodes**
	Small instances (G525,G527,G725)	*Medium instances* (G529,G727,G729)	*Large instances* (G925, G927, G929)	*All instances*
B[ ]	**1.0**	**2.6**	2.7	4.75
B[R1]	2.3	4.0	5.0	3.75
B[R1+S]	4.7	3.3	2.3	**1.5**
B[R2]	4.3	4.3	4.3	4.25
B[R2+S]	5.0	3.7	**1.3**	2.5
B[S]	3.7	3.0	5.3	4.75

Table 11.3: Symmetry testing Basic. The table shows the average ranking of all the combinations of SBCs based on run time on small, medium and large instances. For each instance the SBC combinations have been ranked from 1 (fastest)- 6 (slowest).

Large instances. For the Small and Medium category the Basic model without any SBCs has the best on average ranking. However, on the Large instances category, B[R2+S], stands out from the other ones with a ranking of 1.3. Considering the average run time ranking, B[ ] and B[R1] are getting strictly worse when the instance size grows. B[R2] has the same ranking for all categories, while B[R1] and B[S] have no clear trend. Only B[R1+S] and B[R2+S] have a strictly improving ranking when the instances grow. B[R2+S] perform better than B[R1+S] on the Medium category. On the other hand, B[R1+S] on average has less number of nodes in the B&B tree than B[R2+S]. It is not obvious which of B[R1+S] and B[R2+S] that performs the best. However, since the run time is what matters after all, we chose to include the symmetry breaking constraints R2 and S, because these gave the best run times on the large instances.

As was discussed in Section 9.1 the models suffer a lot from symmetry. Throughout the initial testing of the computational study the fact that symmetry is indeed a great problem has been steadily confirmed. An example is shown in Figure 11.1. It shows the Gap/Run time graph for instance G[9211] on models B[ ] and B[R2+S]. The two models find optimal solution at about the same time, however B[ ] does not manage to prove optimality within the cut off time og 10'000 s. This is probably due to symmetrical solutions, and indicates that it is beneficial to include these symmetry breaking constraints in the model.

Testing of Flower specific symmetry breaking constraints

In Section 9.1.2 one set of Flower-specific symmetry breaking constraints is presented; the loop numbering constraints (L). The symmetry testing on the Flower model indicates good effect of this set of constraints, and it is thus included in the further testing of the Flower model. Table

Average ranking:	**Run time**			**# of nodes**
	Small instances (G524,G525, G526)	*Medium instances* (G527,G529,G5211)	*Large instances* (G5215, G725, G728)	*All instances*
C[ ]	**1.7**	2.0	3.3	4.0
C[T]	2.3	4.3	3.3	4.6
C[O]	4.0	3.3	4.4	3.8
C[U]	3.0	5.7	3.3	3.7
C[TO]	4.7	**1.7**	**1.3**	**2.1**
C[TOU]	5.3	4.0	5.3	**2.9**

Table 11.4: Symmetry testing Chain. Average ranking of the tested combinations of Chain specific SBCs based on run time on small, medium and large instances. For each instance the SBC combinations have been ranked from 1 (fastest)- 6 (slowest). It also shows the average ranking of the # of nodes.

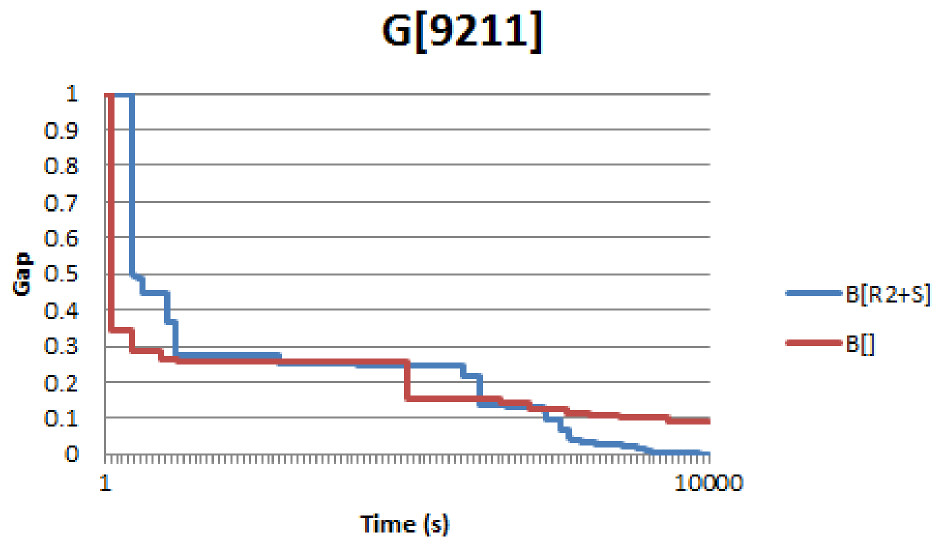


Figure 11.1: The Gap/Time graph for instance G[9211] on models G[ ] and G[R2+S]. The two models find optimal solution at about the same time, however B[] does not manage to prove optimality within the cut off time of 10'000 s. This is probably due to symmetrical solutions.

B.3 shows the results from the symmetry testing of Flower. From this it can be seen that 6 of 9 instances get lower run time by including the *Loop numbering* valid inequality. As much as 8 of 9 instances get a lower number of nodes by including the constraints. To us this seems like a good indication that it will be beneficial to include this cut and that it will have good effect on larger instances too.

Testing of Chain specific symmetry breaking constraints

In Section 9.1.3 three sets of Chain-specific SBCs were proposed; namely *Top layer priority* (T), *One layer for simple routes* (O) and *Restricted twin arcs usage* (U). In this Section the outcome of the testing is presented. The complete overview of the test results can be found in Appendix B.3.

The testing was performed as follows: First the three sets of constraints were added separately and run on nine instances of type General with a varying number of ports and demands. This initial testing showed that the performance of T and O was significantly better than for U, as can be seen from Table 11.4. Then the same instances were run on Chain model with T and O in combination, i.e. C[TO], and the Chain model included all three SBCs, i.e. C[TOU], to see if any of these combinations outperformed the SBCs in isolation. The results of the testing indicate that it will be beneficial for the technical performance of the Chain model to include the two first mentioned sets of SBCs, i.e. T and O.

11.3 Evaluation of valid inequalities

In this section the results from the testing of the different valid inequalities are presented. The effect of adding strong valid inequalities is twofold. On one hand a tighter formulation results in a smaller branch-and-bound tree which is desirable with respect to the run time. On the other hand, this comes at the cost of the added complexity due to an increased number of variables and constraints. In this section we investigate which of the two effects that dominates the other for the valid inequalities that were proposed in Chapter 9 - to see which of them that should be added to the formulations.

Testing procedure. Figure 11.2 illustrates the procedure of which the testing of the valid inequalities was performed. First, we performed testing to investigate if the Xpress presolve should be enabled or disabled. Then the general valid inequalities were tested. The first decision was to determine which α-values and -combinations that seemed best for the integer rounding valid inequalities. After that, the general valid inequalities were tested both separately and in combinations. The ones that performed well, were added to the models in advance of the testing of the Flower and Chain specific valid inequalities.

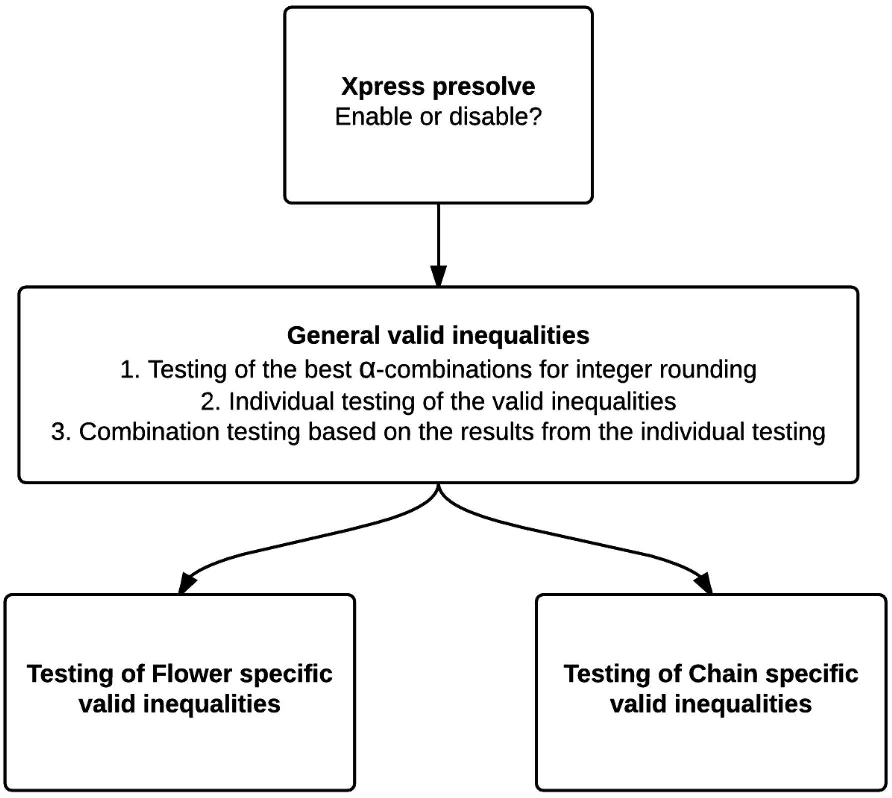
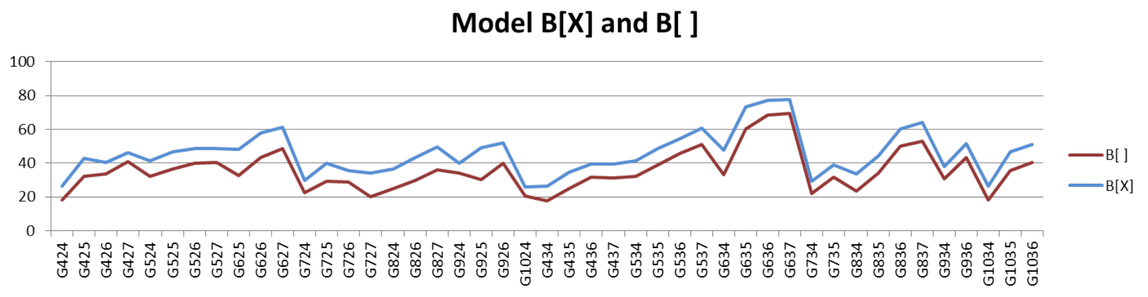

Xpress presolve
Enable or disable?
General valid inequalities
1. Testing of the best α-combinations for integer rounding
2. Individual testing of the valid inequalities
3. Combination testing based on the results from the individual testing
Testing of Flower specific valid inequalities
Testing of Chain specific valid inequalities

Figure 11.2: Testing procedure of valid inequalities

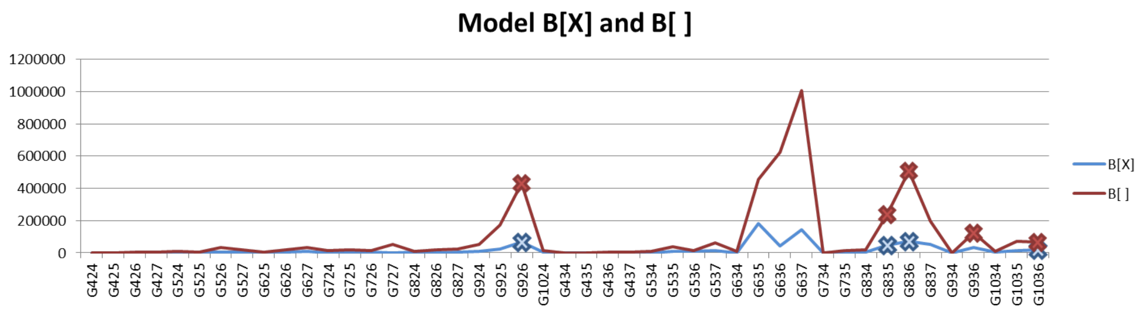
11.3.1 Testing Xpress presolve

Xpress Optimizer's presolve function generates sophisticated valid inequalities and improves the bounds of the problem before the actual branch-and-bound search starts. In general, disabling presolve should lead to a larger search tree and worse upper and lower bounds. However, even though the presolve facility may improve performance by modifying the user's matrix so that it is easier to solve, it also adds complexity in the problem, that can influence the run time in the opposite direction.

When testing the different valid inequalities, the models have been run with presolve disabled, as well as enabled, in order to test the effect of valid inequalities without having Xpress Optimizer generate improvements of its own. The intention was to figure out if the rest of the testing should be run with or without Xpress presolve.



(a) The LP objective values from the runs of B[X] and B[ ] on 45 instances



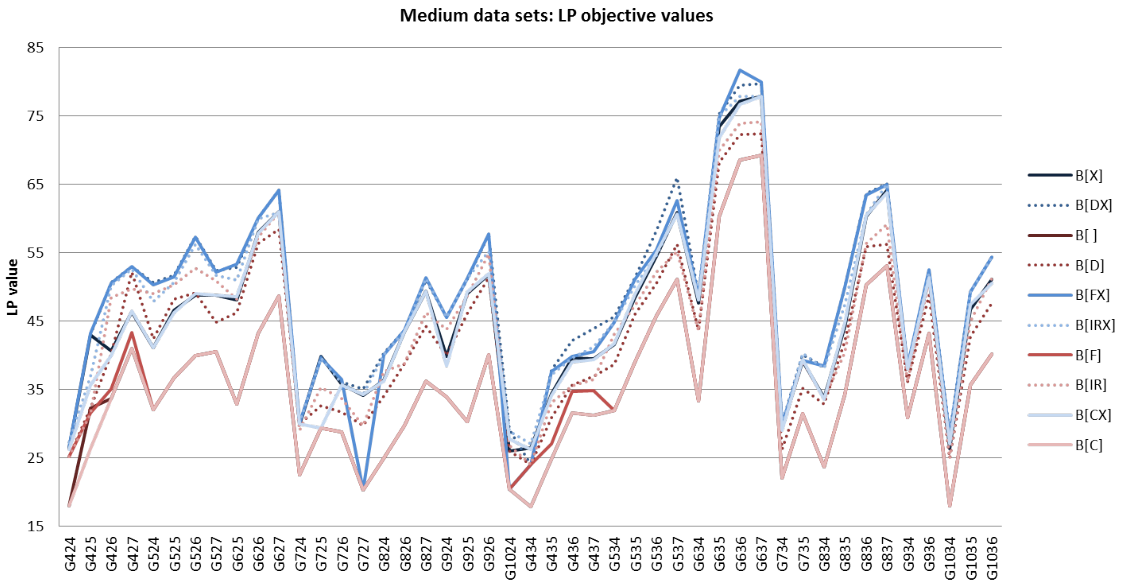
(b) The number of nodes from the runs of B[X] and B[ ] on 45 instances.

Figure 11.3: Results from testing the Xpress presolve

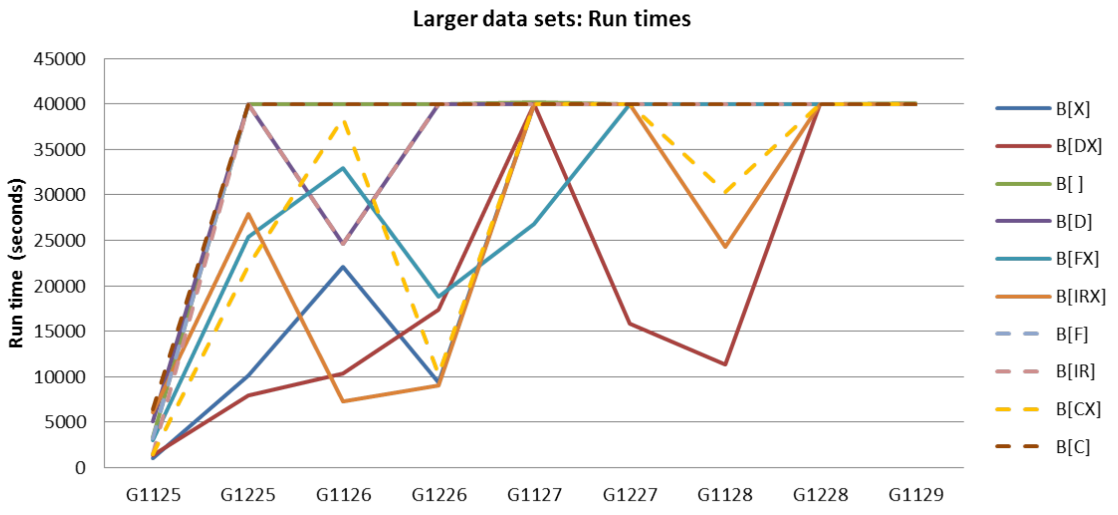
In order to decide if the Xpress presolve should be turned on, 45 instances were run on the B[ ], B[C], B[D], B[F] and B[IR], and the corresponding models with Xpress enabled. For all runs (45*5=225) the LP objective value improved, i.e. increased, between 12% and 68%. Figure 11.3a shows the LP objective value of B[ ] and B[X] on all 45 runs. It shows clearly that the LP objective value of B[X] is consistently better than for B[ ].

The testing also shows that the number of nodes are lower for B[X] than B[ ], in particular for large and more complex inputs. While the number of nodes increase drastically for B[ ], it is severely more modest for B[X]. This tendency is apparent in Figure B.8a. The results from testing reveals that the run times for the models with Xpress presolve enabled in general are equal to or better than the models without, especially for the largest inputs. All the results (LP objective values, # of nodes and run times) can be found in in Appendix B.2.

From Figure we observe that B[ ], B[F] and B[C] only manage to solve the first data input within the time limit. B[X], B[DX], B[IRX], B[CX] and B[FX] have better results and appear at the lower parts of the graph, which means that they for several of the inputs manage to solve to problem to optimality within the time limit.



(a) The LP objective values of all the models including different valid inequalities.



(b) The run times the Basic model with different valid inequalities with larger data inputs

To summarize; the testing showed that the Xpress presolve improves the performance significantly, especially with regards to the LP objective value. Thus, the Xpress presolve was enabled for all consecutive runs in the computational study.

11.3.2 Finding good α-parameters for integer rounding

As described in Section 9.3 the values of the α parameters have to be calculated in advance to derive the integer rounding valid inequalities. The best α parameters are those that make the left hand side of the inequality to be just above some integer, β.

Models B[C] and B[IR] were tested with the following values and combinations of β (ϵ was set to 0.001 in all tests):

- **Alt. 1:** $\beta = 1$
- **Alt. 2:** $\beta_1 = 1$ and $\beta_2 = 2$
- **Alt. 3:** $\beta_1 = 1$, $\beta_2 = 2$ and $\beta_3 = 3$

To set the α-values for the integer rounding cuts, we followed the procedure described in Section 9.3. By looking at which values and combinations that gave the best effect in terms of LP objective values, number of nodes and run time the α-decisions were made. For B[CX] Alternative 3 performed the best, while for B[IRX] Alternative 1 seemed to be the best. A detailed description of the results, and how they were analyzed, is given in Table B.4.

11.3.3 Results from separate testing of the general valid inequalities

The purpose of the individual testing was to get an impression of the effect of each set of general valid inequalities, and use this to determine which combinations of valid inequalities we should combine in further testing. By comparing the results from the runs of B[X] with and without including a particular set of valid inequalities, their isolated effect were investigated.

The testing of the valid inequalities was performed on the same 45 instances as the testing of the Xpress presolve, as well as 9 larger instances. The first 45 had a cut off time of 10’000 seconds (3 hours), while the large 9 instances the cut off time was set to 40’000 seconds.. All the results from the testing can be found in Appendix B.2.

Table 11.3.3 shows a summary of the observed performance of the testing of the general valid inequalities based on the runs on 54 instances. The instances were categorized into Small,

Performance of valid inequalities

🟩	= Significantly improved
🟨	= Slightly improved
🟥	= Equal or worse

Valid inequalities	LP. obj value			# of nodes			Run time		
	Small	*Medium*	*Large*	*Small*	*Medium*	*Large*	*Small*	*Medium*	*Large*
B[CX]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
B[DX]	🟩	🟨	🟨	🟨	🟩	🟩	🟥	🟨	🟩
B[FX]	🟩	🟥	🟨	🟨	🟩	🟨	🟨	🟩	🟨
B[IRX]	🟨	🟨	🟩	🟨	🟩	🟩	🟥	🟩	🟨

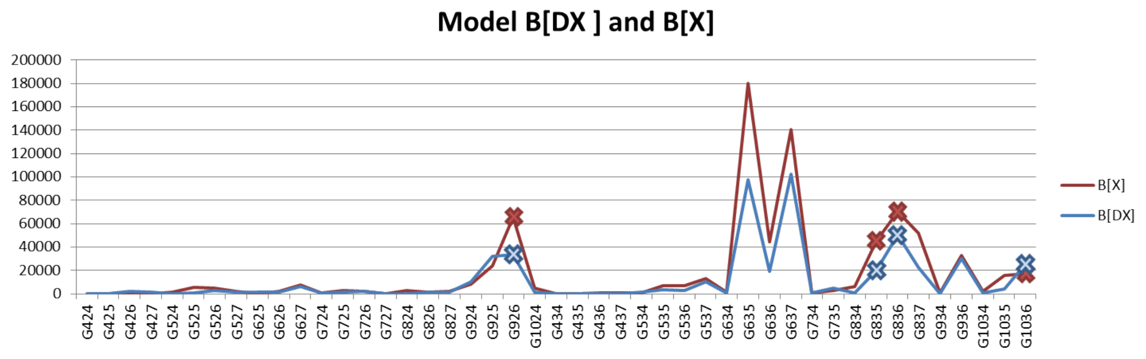
Table 11.5: Individual performance of valid inequalities. The table shows how the performance of the Basic model changed by adding the general valid inequalities. The assessment is based on the runs of 54 instances. The instances and their categorization can be found in Table B.6.

Medium and Large instances, to be able to see how the model scaled with the different valid inequalities. An overview of the instances applied in the testing, as well as their categorization is given in Table B.6.

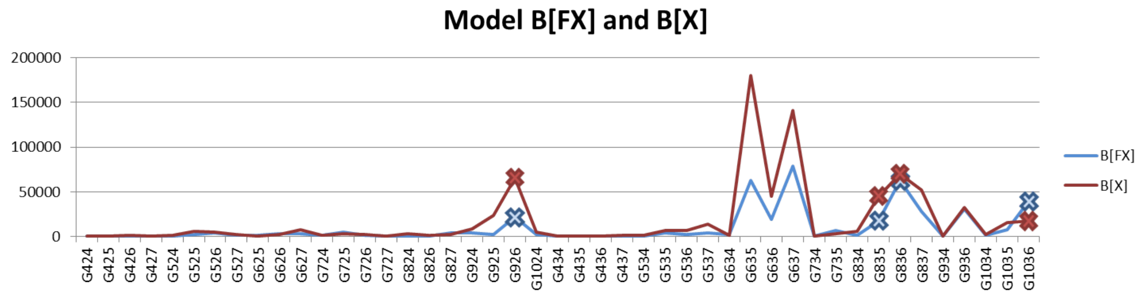
From Table 11.3.3 it is apparent that the Capacity miles cut with integer rounding scores very poorly on every point of measure. The three other sets of general valid inequalities, on the other hand, have a positive effect.

Particularly the number of nodes in the B&B tree show a significant decrease in B[DX] and B[IRX] compared to B[X]. The decline gets relatively stronger for the larger instances. This tendency for the D/C valid inequality can be seen in Figure 11.5a.

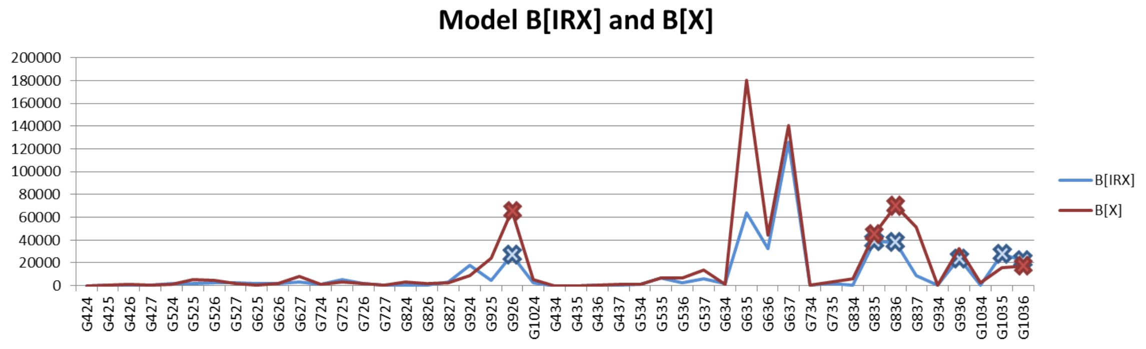
B[DX] shows good performance in run times, especially on the large instances. This can be seen from Figure 11.4b.



(a) The number of nodes on B[DX] and B[X] from runs on 45 different instances



(b) The number of nodes on B[FX] and B[X] from runs on 45 different instances



(c) The number of nodes on B[IRX] and B[X] from runs on 45 different instances

Figure 11.5: The graphs show how the number of nodes is influences in the runs on 45 instances by including the general valid inequalities D, F and IR. It is clear that the nodes are reduced, especially for the large and complex instances.

11.3.4 Results from testing combinations of valid inequalities

The testing of each valid inequality presented above, showed the best results for B[DX], second best for B[IRX] and third best for B[FX], while the Capacity Miles-valid inequality only made the Basic model perform worse. In order to see if any of the sets of valid inequalities in conjunction would give better results than any of them separately, we run tests on combinations of them. Since running tests on all combinations would be too time consuming, we chose to combine the best performer (B[DX]) with all the others, the second best with all the others but the worst performer (B[FX]), etc. The combinations of valid inequalities that were tested are shown in Table 11.6. We compared the combinations with B[DX] since this showed the best effect separately.

	Valid inequality combinations		
1	B[DX]	⇒	B[DX]
2	B[DX] and B[IRX]	⇒	B[DIRX]
3	B[DX], B[IRX] and B[FX]	⇒	B[DIRFX]
4	B[DX], B[IRX], B[FX] and B[CX]	⇒	B[DIRFCX]
5	B[DX] and B[FX]	⇒	B[DFX]
6	B[DX] and B[CX]	⇒	B[DCX]
7	B[DX], B[IRX] and B[CX]	⇒	B[DIRCX]
8	B[DX], B[FX] and B[CX]	⇒	B[DFCX]

Table 11.6: The combinations of valid inequalities that were tested. The combinations were based on the results from the individual testing of the valid inequalities.

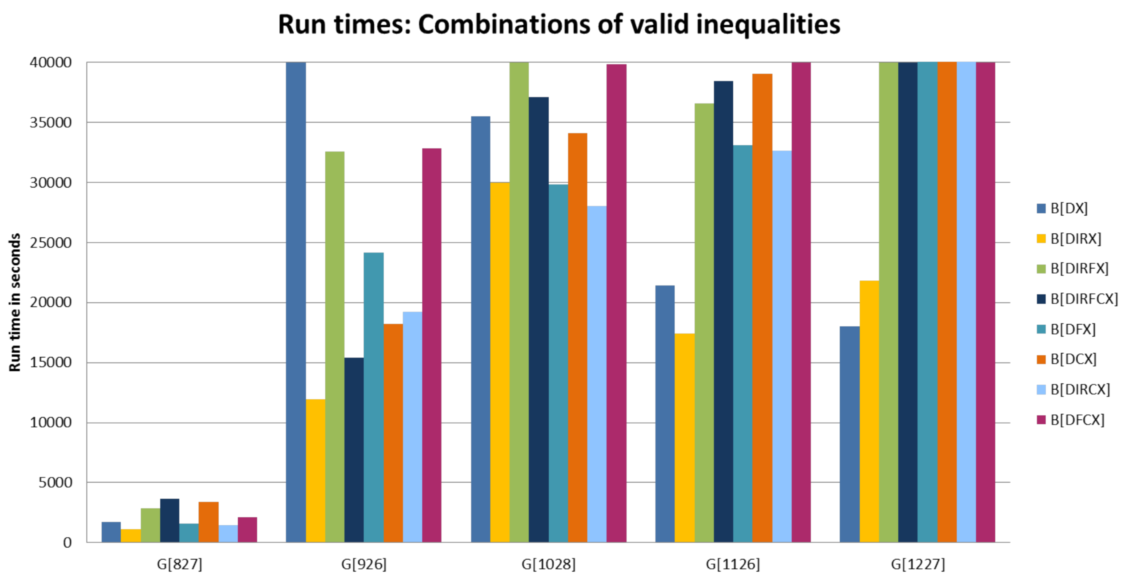


Figure 11.6: Run times of the valid inequality combinations on instances G[827], G[926] and G[1227].

Figure 11.6 shows the run times of the Basic model with the 8 combinations of valid inequalities on five large instances. Model B[DIRX] has the best run time in 3 of these, while on the two others it is the second and third best performer. From the tests we have done B[DIRX] seems to have the best combination of valid inequalities.

To sum up; the following general valid inequalities are added to all models: *Simple Demand/Capacity*(D/C)- valid inequality and the *Simple Demand/Capacity*(D/C)- valid inequality with integer rounding.

Evaluation of Flower specific valid inequalities

In Section 9.2.2 one set of Flower specific valid inequalities is introduced; namely the *Only one first/last loop* (N)-valid inequalities given by Constraints (9.21)-(9.24).

In Table 11.3.4 the results from the testing of the *Only one first/last loop*-valid inequalities are presented. It shows the difference in the number of nodes and the run time for 9 different instances, run on F[X] and F[XN]. The number of nodes are mostly lower for the model variant excluding the valid inequalities, F[X], than the one including them, F[XN]. The run time is better for F[XN] than for F[X] in 6 out of 9 test instances. We can see no pattern in which is better at the more complex and larger instances, however F[X] had significantly lower run time (1960.69 vs. 2965.92) than F[XN] on the largest instance tested, G[1125] When it comes to the LP objective values no improvements were observed.

Even though the run time results might account for the "Only one first/last loop"-valid inequalities to be included, the number of nodes results and insignificant improvement in LP objective value contribute to the conclusion of not including the valid Flower specific valid inequalities.

Results of Flower specific valid inequalities

🟩 = Lowest # of nodes/run time for given instance

	# of nodes		Run time (s)	
	F[X]	F[XN]	F[X]	F[XN]
G[425]	🟩 365	725	25.07	🟩 15.81
G[437]	🟩 441	1249	40.10	🟩 34.30
G[526]	🟩 5591	9763	🟩 100.96	155.17
G[537]	🟩 6107	14149	🟩 158.26	194.23
G[634]	🟩 1965	2917	155.74	🟩 139.75
G[735]	6103	🟩 1937	354.57	🟩 229.03
G[827]	🟩 2761	4311	683.74	🟩 637.24
G[926]	35953	🟩 35679	2172.08	🟩 1753.72
G[1125]	🟩 9435	23965	🟩 1960.69	2965.92

Table 11.7: Results from testing Flower specific valid inequalities. The number of nodes and run times from the runs of F[X] and F[XN] on 9 instances. The cut off time was set to 10'000 seconds on all runs.

Evaluating Chain specific valid inequalities

One Chain specific set of valid inequalities has been proposed, namely the *Two single visits* (V)-valid inequalities, given by constraints (9.25).

The test results from testing of the Chain model specific valid inequalities are shown in Table 11.3.4. As can be seen from the table, V seems to reduce the number of nodes on the larger data instances.

The run times are also better for C[XN] for all inputs except G[827], and for this data instance the difference is insignificant (the run time for C[X] is 640 seconds, or only 3% better than C[XV]'s run time). The results showed that the LP objective values were not improved significantly, but they were slightly improved for the majority of the instances.

Neither C[X] nor C[XV] managed to solve instance G[12212] within the cut off time of 10'000 seconds (3 hours). However, C[XV] managed to find an integer feasible solution, as opposed to B[X].

Although the results are not completely indisputable, we chose to include the V-valid inequality. The strongest arguments were that B[XV] showed good performance on the larger instances and managed to find an integer solution, when its counterpart without the cut did not.

Results of Chain specific valid inequalities

🟩 = Lowest # of nodes/run time for given instance

🟥 = Did not prove optimality within cut off time. Gap shown if integer solution found.

	# of nodes		Run time (s)	
	C[X]	C[XV]	C[X]	C[XV]
G[425]	🟩 3441	5849	60.96	🟩 33.12
G[437]	🟩 9479	10529	324.38	🟩 280.09
G[526]	19221	🟩 10665	338.08	🟩 204.33
G[537]	🟩 44956	50127	3929.52	🟩 3073.28
G[538]	38027	🟩 25531	3591.38	🟩 3348.16
G[736]	49007	🟩 28181	4217.26	🟩 4170.85
G[827]	44651	🟩 27869	🟩 23377.5	24017.8
G[828]			🟥 17 %	🟥 9 %
G[12212]			🟥 -	🟥 21 %

Table 11.8: Results from testing Chain specific valid inequalities. The number of nodes and run times from the runs of C[X] and C[XV] on 9 instances. The cut off time was set to 10'000 seconds on all runs.

Chapter 12

Computational study

This chapter provides results and analyses from the computational study of the six models presented in Chapters 5 to 8.

The models now includes the chosen strengthened formulations according to the results presented in Chapter 11. Table 12.1 summarizes the results from the evaluation of strengthened formulations. For the corresponding models with and without transit time, the same combinations of SBC's and valid inequalities are chosen. The models are referred to as Basic, Flower, Chain, Basic TT, Flower TT and Chain TT, and the interpretation of this is the models with the strengthening formulations according to Table 12.1.

The aim of the computational study is to evaluate the performance of the models considering the solutions they produce as well as getting insight into their computational efficiency, i.e. run time and scalability.

The test instances used in the computational studies, and how they are generated, are described in Chapter 10. The LS-NDP problems solved to optimality are comparable in size to the test instances presented in recent literature on shipping network design using exact methods (e.g. Alvarez (2009) and Reinhardt and Pisinger (2011)). The cut off times differ among the runs, and are explicitly stated when referring to specific results.

The computational studies are based on implementation of the original models and transit time models in commercial optimization software. Xpress Mosel has been the modeling language. The implementations have been solved using Xpress 1.24.00 64 bit with Xpress Optimizer Version 24.01.04. The code was run for different test instances, which are introduced in Section 6.1. All runs were performed on a cluster of computers with HP bl685c G7, 4 x AMD Opteron 6274 2.2GHz, 128 GB of RAM, and 300 GB SAS 15000 rpm.

In Section 12.1 the impact of the size of the route set, $\mathcal{R}$, is discussed. There is also a discussion of how it should be set. Section 12.2 provides results and analyses of the computational study of the original models, while in Section the same is presented for the transit time models. In both sections technical and economical results are discussed.

		Basic and Basic TT	Flower and Flower TT	Chain and Chain TT
SBCs				
All models:	R1	✗	✗	✗
	R2	✓	✓	✓
	S	✓	✓	✓
Flower specific:	L	N/A	✓	N/A
Chain specific:	T	N/A	N/A	✓
	O	N/A	N/A	✓
	U	N/A	N/A	✗
Valid inequalities				
All models:	C	✗	✗	✗
	D	✓	✓	✓
	F	✗	✗	✗
	IR	✓	✓	✓
	X	✓	✓	✓
Flower specific:	N	N/A	✗	N/A
Chain specific:	V	N/A	N/A	✓

Table 12.1: Overview of included symmetry breaking constraints and valid inequalities in the six models.

12.1 Impact of the size of the route set

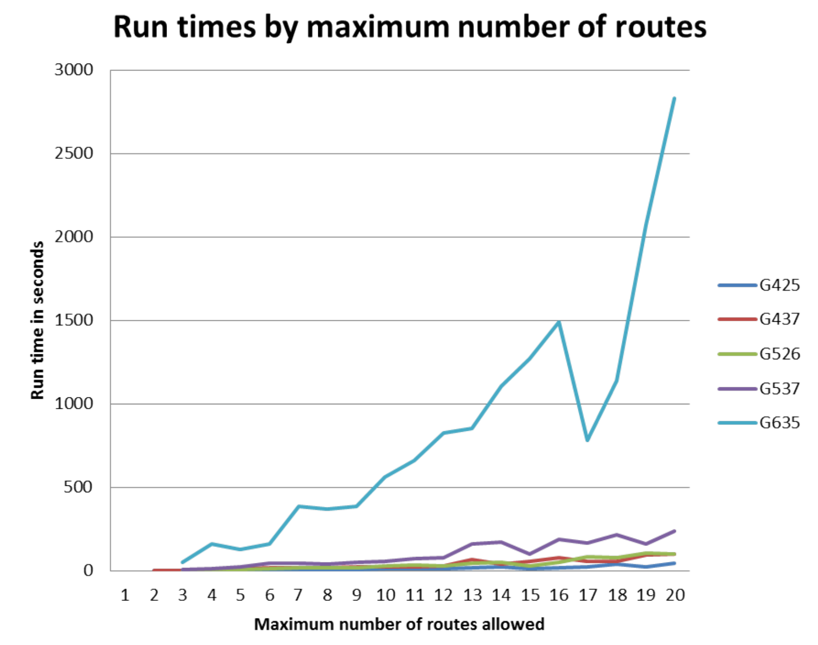
In this section we investigate and discuss the impact of the route set size, on the computational efficiency of the models. We also explain the rationale behind how the route set size was set in this computational study.

Recall that the models have a predefined set, $\mathcal{R}$, of route indexes available. This is necessary because we do not not in advance how many routes there will be in the optimal solution. As have been discussed in previous chapters, this causes a lot of symmetry. Even though most of the symmetry is removed by the symmetry breaking constraints, the size of the $\mathcal{R}$ still has a strong correlation with the run time. This is because almost all variables are indexed with r, and most of the sets of constraints are per route. In the preliminary testing of the strengthened formulations the size of the route set was set to 20. Through testing we observed that the size of the route set clearly affects the run time. This can be seen very clearly from Figure 12.1. Thus, considering the computational efficiency of the models, there is a strong incentive to not have a route index set size that is larger than necessary.

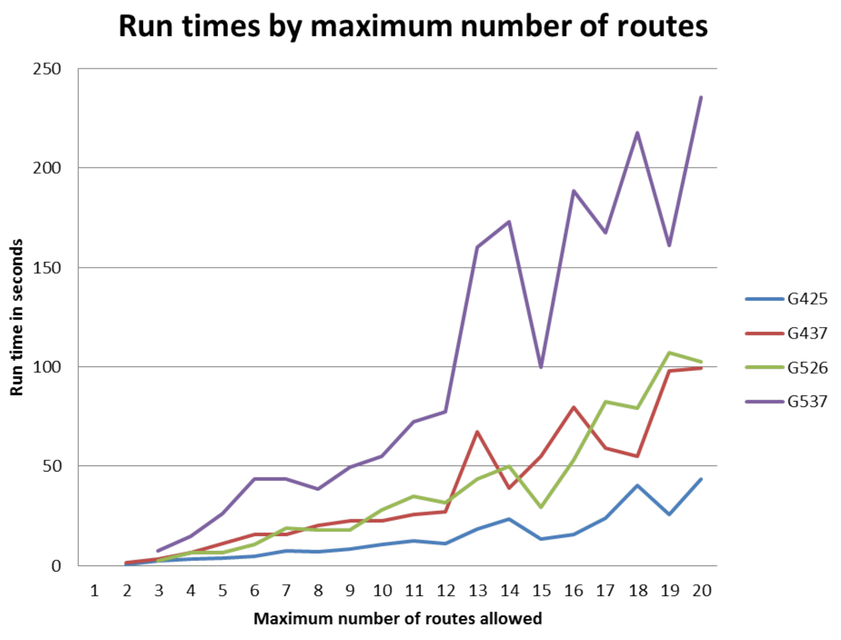
On the other hand, the maximal number of routes should not be at the expense of finding the optimal, or good feasible, solutions. Consequently, the challenge is to find an optimal size of the route set that balance these two objectives. To investigate how the objective values change along with the available number of route indices, we run tests on a sample of instances with $|R|$ from 1 to 20. The results are shown in Table 12.2. Instance G[1329] has the optimal number of routes of 5, while for the other instances the number is lower. The largest number of routes observed in the overall testing has been 6.

In further testing, we chose to set the route set size to 9. This way we could save some time, be able to do more testing, and hopefully not miss any optimal solutions.

For further use of the models there could be more sophisticated ways of finding a reasonable route set size. Perhaps, the size could be set as a function of the number of ports, demands and ship types, or as the number of routes in the LP solutions. There are plenty of ways, but this will also depend on the size and type of instances, so we will leave this for the users to decide.



(a)



(b)

Figure 12.1: How run times vary with the route set size

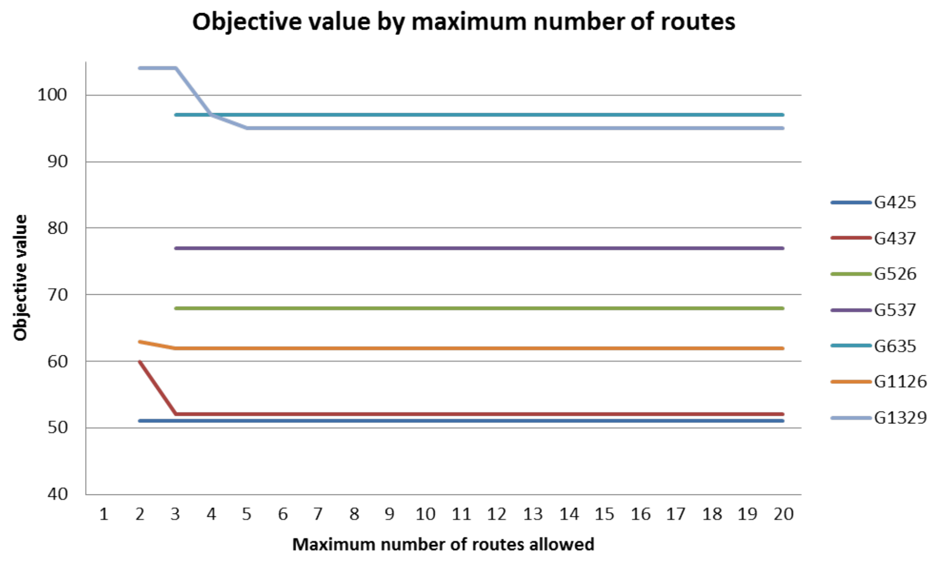


Figure 12.2: Objective values by size of route set

12.2 Original models

In this section, the computational results and economical solutions of the original models (Basic, Flower and Chain), are analyzed and discussed.

First, in section 12.2.1, the technical results are presented. Secondly, in Section 12.3.2, the solutions of the three models are analyzed and compared.

12.2.1 Technical results

In this section, the technical results from the testing of the original models on 32 different data instances, are presented. Of the data instances 16 of them were of a General network type, while there were 8 instances of both the Feeder and Hub & Spoke network type.

On all runs the route set size, $|\mathcal{R}|$, is set to 9, like stated in Section 12.1. The upper run time limit is set to 40'000 seconds.

Run times, best integer solutions and gaps

The run times range from 14 seconds to the upper run time limit of 40'000 seconds. In order to show the nuances in the difference between the models' run times, the run time results are presented in three different graphs. Figure 12.3 shows the run times for the three original models for the results where all the run times are up to 800 seconds. Figure 12.4 shows the results of when the highest run time is between 800 seconds and less than 25'000 seconds. And finally, table 12.2 show the run times, the best integer solutions and the gap where at least one of the models have a run time of more than 25'000 seconds, which also turned out to be when at least one of the models did not solve to optimality within the run time of 40'000 seconds.

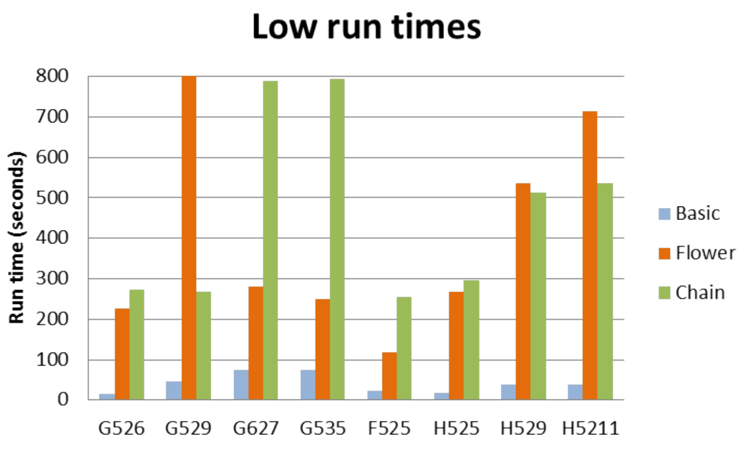


Figure 12.3: Run times of the models, where all run times are under 800 seconds.

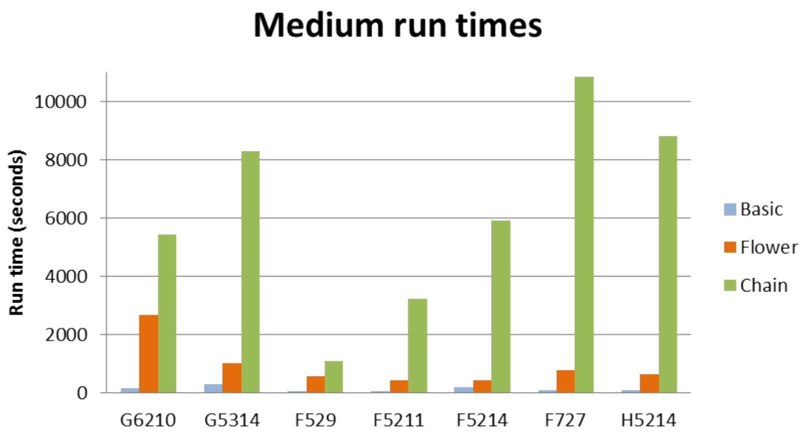


Figure 12.4: Run times of the models, where all the run times are under 25'000 seconds.

Figure 12.3 shows that the Basic model clearly has the lowest run times for the instances that all the models manage to solve within a run time of 800 seconds. For four of the data inputs, the Flower model has a higher run time than the Chain model, and for five of them, it is the other way around. The common feature for the inputs where the Flower model perform better than the Chain model is that there are few demands (7 or less). For the inputs where there are more demands (G[529], H[529], H[5211], H[215]), the Chain model manage to solve the problem faster.

Figure 12.4 shows the results for the instances that were solved within the medium run time (25'000).

The results show that Chain's run times are significantly larger than for the two other models.

	Basic			Flower			Chain		
	R.t.	**Best int. sol.**	**Gap**	**R.t.**	**Best int. sol.**	**Gap**	**R.t.**	**Best int. sol.**	**Gap**
G[**829**]	1380.62	125	0 %	-	*135*	15 %	-	*121*	22 %
G[**9213**]	-	*176*	14 %	-	*202*	51 %	-	*259*	93 %
G[**10214**]	-	*191*	28 %	-	*306*	123 %	-	*237*	70 %
G[**12212**]	-	*161*	28 %	-	*	*	-	*	*
G[**15213**]	-	*222*	71 %	-	*	*	-	*	*
G[**15215**]	-	*308*	124 %	-	*	*	-	*	*
G[**7311**]	3951.56	105	0 %	26004.3	96	0 %	-	*204*	153 %
G[**7314**]	5403.27	117	0 %	-	*117*	3 %	-	*155*	71 %
G[**9311**]	-	*130*	13 %	-	*154*	60 %	-	*	*
G[**9314**]	-	*191*	65 %	-	*177*	62 %	-	*	*
F[**7211**]	1688.81	132	0 %	18007.7	132	0 %	-	*141*	27 %
F[**7213**]	4849.89	148	0 %	-	*156*	21 %	-	*150*	19 %
F[**7215**]	5142.41	158	0 %	-	*160*	14 %	-	*166*	28 %
H[**727**]	1674.77	308	0 %	-	*320*	30 %	-	*286*	2 %
H[**7211**]	-	*357*	5 %	-	*378*	47 %	-	*	*
H[**7213**]	-	*	*	-	*	*	-	*	*
H[**7215**]	-	*495*	20 %	-	*	*	-	*	*

Table 12.2: Run time, best integer solution and gap from testing of the original models for data inputs where at least one model did not solve to optimality within the run time limit. * = No results due to no found integer feasible solution. Italic numbers are when the result is not proven optimal but integer feasible. - = Run time limit reached.

The run times of Flower are also a lot larger than for Basic, but the difference is not as huge as between Chain and the other two. For six out of seven instances the run time of Chain is more than twice the run time of Flower. For all data inputs except F[529], the run times of Chain are higher than 3'200 seconds, while both the Flower model and the Basic model manages to solve the problem in a run time beneath 2'700 seconds. Common for all the data inputs in Figure 12.4 is that the number of demands are high, ranging from 7 to 14.

The results of the data inputs that the models have spent the longest time solving, or not managed to solve at all, are presented in Table 12.2. For 10 of the data inputs, neither of models manage to solve to optimality within 40'000 seconds. For 5 of the data inputs (G[829], G[7314], F[7213], F[7215] and H[727]), the Basic model is the only model that manages to solve to optimality within the run time limit. There are also two instances, G[7311] and F[7211], where both the Basic model and the Flower model manage to solve to optimality, while the Chain model does not. However, for two of the data inputs (G[829] and H[727]), the Chain model's best bounds and feasible integer solutions are not proven optimal, but are still better than the Basic model's and the Flower model's. For H[727], the gap is only 2%, so the solution found is likely to be the optimal one. This is also the case for the Flower model for data input G[7314] where the Flower model has found an equally good solution as the Basic model, and the gap is only 3%, and for G[9314] the best bound is not proven optimal, but better than the Basic model's solution.

From the results shown in graphs in Figure 12.3, Figure 12.4 and Table 12.2, it is clear that the Basic model has the shortest run times. This is not surprising, being the least complex model, the Basic model has less variables and less constraints. The differences between the Chain model and the Flower model vary more, but while the Flower model's run time outperform the Chain model's for 14 of the data inputs, the Chain model only have a better run time for 3 of the data inputs. Looking only at the results in Figure 12.3, the Flower model and the Chain model seem to perform equally good, but for the more challenging instances (where the general solution time is high), Chain's run times are always higher than Flower's. Even though Chain and Flower do not manage to prove optimality on some instances, they still manage to find better integer solutions than Basic's proven optimal solution, within the cutoff time.

Number of nodes in B&B tree

The number of nodes is another feature of the models that is interesting to explore in a technical analysis. The number of nodes says something about the size of the B&B tree and the integrality features of the problem. More integer variables generally lead to more nodes in the tree. Table 12.3 shows the number of nodes for the instances that all three models managed to solve to optimality. For each data input, the model with the lowest number of nodes are marked green, and the largest number of nodes are marked red.

	Basic	**Flower**	**Chain**
G[526]	911	8261	9447
G[529]	2987	5415	1647
G[627]	5595	1467	15155
G[6210]	6451	51031	31446
G[535]	223	1661	11264
G[5314]	7715	4429	38034
F[525]	707	1923	12999
F[529]	317	5591	27525
F[5211]	2241	11253	39827
F[5214]	30521	8977	79551
F[727]	8205	3813	101872
H[525]	1353	17543	12733
H[529]	1077	26757	7584
H[5211]	495	19513	12563
H[5214]	14923	25809	153601

Table 12.3: Number of nodes from testing of the original models

In most cases, the Basic model has the fewest numbers of nodes and the Chain model has the highest number of nodes. For the Flower model, the number of nodes varies mores. For some of instances, the Flower model has the fewest number of nodes, for some it is in the middle, and for others, it has the most. The Chain model's high number of nodes could possibly be caused by the doubled network. An additional port in the Chain model has more severe consequences than for the other models. For instance: In the Basic model and the Flower model, one additional port means 2(N-1) additional arcs (an arc back and forth from the new node to the existing nodes). However, this is different for the Chain model: In the Chain model, one additional port added will create as much as 4(N-1) + 2 new arcs (2(N-1) (arcs between ports) + 2(N-1) (arcs between twin ports) + 2 (arcs between port and twin port). This will also create a lot of symmetric solutions. Some of these are removed with the symmetry breaking constraints, but there are still symmetric solutions in the solution space caused by the doubled network.

Average solving time per node

Another feature that may say something about the technical properties of a model is how long time it takes to solve the nodes of the B&B tree. In each node in the B&B tree the LP-relaxation is solved and thus the time spent in a node is related to solving the LP-relaxation. The time spent on solving each node therefore indicates the complexity of the problem not related to its integrality features.

Table 12.4 shows the average solving time per node $\left(\frac{RunTime}{NumberOfNodes}\right)$.

	Basic	**Flower**	**Chain**
G[526]	0.016	0.027	0.029
G[529]	0.015	0.148	0.162
G[627]	0.013	0.192	0.052
G[6210]	0.022	0.052	0.173
G[535]	0.327	0.150	0.070
G[5314]	0.039	0.227	0.218
F[525]	0.032	0.061	0.020
F[529]	0.146	0.104	0.040
F[5211]	0.029	0.037	0.081
F[5214]	0.006	0.046	0.075
F[727]	0.011	0.203	0.106
H[525]	0.014	0.015	0.023
H[529]	0.036	0.020	0.068
H[5211]	0.077	0.037	0.043
H[5214]	0.005	0.025	0.057
Average	*0.053*	*0.090*	*0.081*

Table 12.4: Average solving time per node

The average solving time in a node is lowest for the Basic model, followed by the Chain model. The average solving time per node in the Flower model is more than twice as high as for the Chain model. This observation makes intuitively sense because the Flower model has many complicated constraints making the structure of the problem difficult, independently of the integrality of the problem. Knowing that the main difference of the Basic model and the Chain model is the Chain model's doubled network, not the constraints and how the problem is solved, it makes sense that the Chain model's average solving time per node are closer to the Basic model's average solving time than the Flower model's average is. The increase in solving time per node for the Chain model compared to the Basic model could possibly be caused by the additional complexity that the Chain model's twin network approach includes regarding demand flow constraints, ship flow constraints, transhipment constraints, and also the Chain specific valid inequalities and symmetry breaking constraints.

12.2.2 Comparison of solutions

In this section, the solutions from the original models are explored and presented. The aim is to show that the models behave as expected, and to compare the solutions they give for different instances.

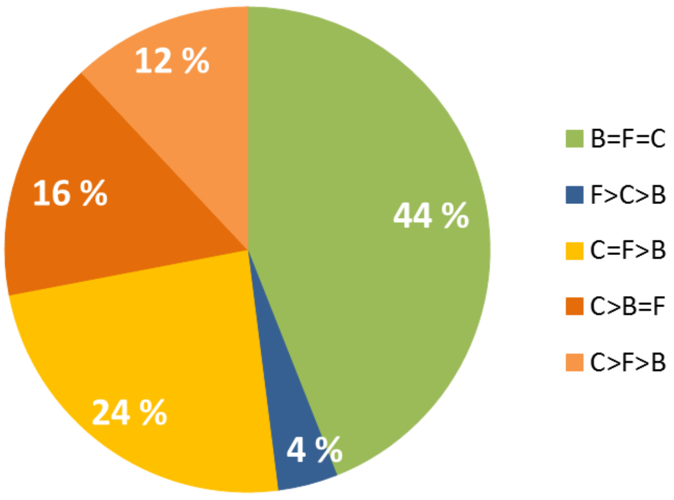
Economical comparisons

Figure 12.5 shows an overview of economical results for original models.

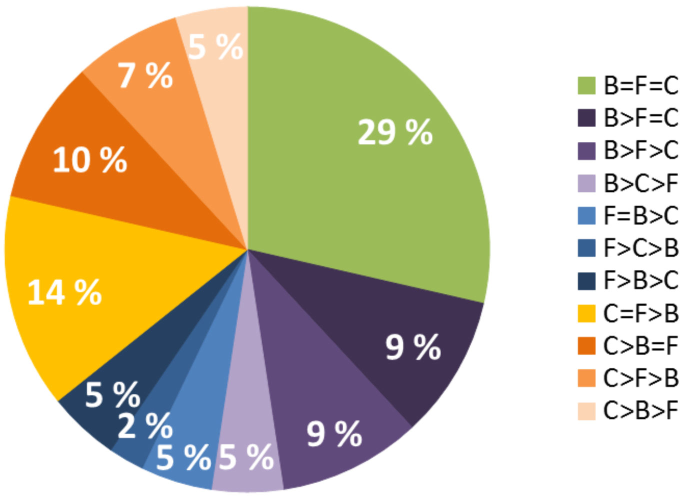
Figure 12.5a shows the frequency of which models most often give the best objective value, based on the instances in the testing where all models managed to solve to optimality. This was 25 of 42 tested instances. For 44% of the instances the three models perform equally well. For 24% of the instances Chain and Flower provide an equally good solution that is better than the one Basic gives. Chain is better than both the others in 28% of the cases. For 57% of these instances Basic and Flower provide the same objective value, while for the remaining 43% Flower outperforms Basic. In 4% (which corresponds to one instance) of the cases, Flower is the better than both the others.

Figure 12.5b shows which model had the best integer solution at cut off time, independently of whether it was proven optimal or not. This gives a whole other picture of the performance of the different models than Figure 12.5a. In 23 % of the cases Basic finds an integer solution better than the best integer solution found by Flower and Chain. This means that neither Flower nor Chain solved those instances to optimality within cut off time. We know this because the whole solution space of Basic is included in Flower and Chain, so if they solve to optimality, Basic will never be able to provide a better solution than the two other models. This is in accordance with what was shown in Figure 12.5a. For 29 % of the instances the three models find an equally good integer solution. This number is lower than for the corresponding value shown in Figure 12.5a.

To sum up the observations; In 65 % of the cases Chain finds the best integer solution after 40'000 seconds. The corresponding numbers for Flower and Basic are 55 % and 56 %, respectively. This means that even though Chain does not prove optimality, it still finds a better solution than Basic after 40'000 seconds.



(a) Overview of objective values of instances solved to optimality



(b) Overview of best integer solutions found within cut off time

Figure 12.5: Overview of economical results for original models. The figure shows how often which models give the best objective value. Graph 12.5a is based on the instances in the testing where all models managed to solve to optimality. This was 25 of the 42 tested instances. Graph 12.5b is based on the all 42 instances in the testing. B/C/F = objective value of Basic/Chain/Flower

As has been discussed above the three models provide the best objective values for different instances. By looking closer at the solutions of the different instances, we attempt to investigate in which situations the models outperform each other and when their shortcomings get manifested. Table 12.5 shows the objective value, the sailing cost, visiting cost and transhipment cost from the solutions of a sample of interesting test instances.

As shown in Figure 12.5b, Basic and Flower find equally good solutions that are better than Basic's solution for 5 % of the cases. This can both be caused by the models finding two different solutions with the same objective value, or that Flower and Chain find exactly the same solution. For the latter case, we know that the solution include at least one butterfly route. This is also the case in some of the examples presented in Table 12.5, where the Chain model and the Basic model find the same optimal solution. For the Hub & spoke instance H[528] Flower and Chain achieve an objective value of 123, while Basic only finds a solution of 128. In this case Chain and Flower provide the same solution, namely a butterfly route, which is one of the route structures they both can make, and Basic cannot. This is also the case F[324] and G[426].

Table 12.5 also shows an interesting case of the objective values of instance G[519]. Flower finds a solution that is better than both Chain's solution and Basic's solution. This means that Flower's optimal solution includes at least one butterfly route with at least 3 loops from a single depot. Chain also allows solutions with three loops, but not on a "Flower"-format.

		Model		
Instance		***Basic***	***Flower***	***Chain***
F[324]	***Objective value***	33	**30**	**30**
	Sailing cost	26	24	24
	Visiting cost	7	6	6
	Transhipment cost	0	0	0
G[426]	***Objective value***	66	**62**	**62**
	Sailing cost	50	48	48
	Visiting cost	16	14	14
	Transhipment cost	0	0	0
H[528]	***Objective value***	128	**123**	**123**
	Sailing cost	96	91	91
	Visiting cost	16	16	16
	Transhipment cost	16	16	16
G[519]	***Objective value***	46	**32**	39
	Sailing cost	25	15	15
	Visiting cost	16	15	15
	Transhipment cost	5	2	9
G[5214]	**Objective value**	349	321	313
	Sailing cost	320	290	290
	Visiting cost	20	17	18
	Transhipment cost	9	14	5

Table 12.5: Examples of economical results from testing of the original models. The best values of the objective values for each instance are highlighted.

Impact of the sailing cost calculation

For data instance H[528], the models find the solutions illustrated in Figure 12.6. Table 12.5 shows that the Flower model and the Chain model manage to reduce the sailing cost compared to the Basic model, while the visiting costs and the transhipment costs stay the same. The output also shows that all three models use one larger and more expensive ship on route 1, and several smaller, cheaper ships on the other routes.

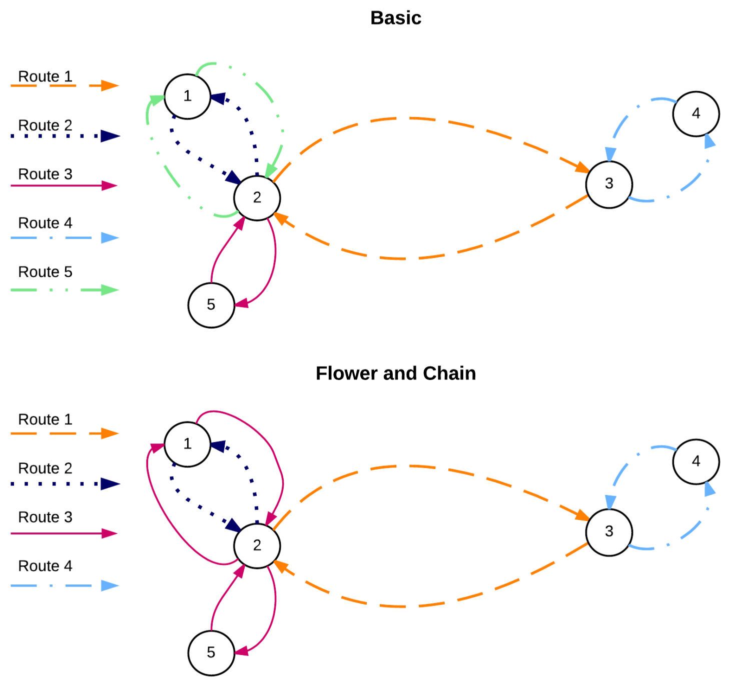


Figure 12.6: Solution network for data instance H[528] with Basic, Flower and Chain.

The reason for this is that the length of a route in weeks (the sailing time) will be rounded up to the nearest integer in order to find the number of ships needed on that route. For instance, if the sailing time of a route is 4.1 weeks, the route needs 5 ships. With two routes with sailing time 3.33 weeks, it thus needs 4 ships. In the Flower model and the Chain model, however, if two routes with sailing time 3.33 weeks are merged into one, it will have a sailing time of 6.66 and only needs 7 ships. Thus, for H[528], the merging of routes will save one ship when the Basic routes are merged into butterfly routes, like the Flower model and the Chain model does. More precisely, route 3 and route 5 in the Basic solution will need 7 ships instead of 8 when they are merged. This is illustrated in Table 12.6.

The advantages of complex routes

The Flower model and the Chain model allow complex route structures, as opposed to the Basic model. Data instance H[5214] resulted in three different solutions for the three original models. The costs are listed in Table 12.5, and the routes generated are illustrated in Figure 12.7.

Basic	**Route length (in weeks)**		**Number of ships**
Route 1 (R1)	2.00	⇒	2
Route 2 (R2)	3.33	⇒	4
Route 3 (R3)	3.33	⇒	4
Route 4 (R4)	3.33	⇒	4
Route 5 (R5)	3.33	⇒	4
	Total number of ships needed		***18***

Flower and Chain	**Route length (in weeks)**		**Number of ships**
Route 1 = Basic R1	2	⇒	2
Route 2 = Basic R2	3.33	⇒	4
Route 3 = Basic R3 + Basic R5	6.66	⇒	7
Route 4 = Basic R4	3.33	⇒	4
	Total number of ships needed		***17***

Table 12.6: Difference in ship utilization in the solutions of Basic, Flower and Chain on instance H[528]

The Basic model generates 5 different routes which results in a visiting cost of 20 and a sailing cost of 320. Including the transhipment cost of 9, the Basic model's solution is the most expensive one. The Flower model's solution includes two butterfly routes (Route 1 and Route 2), which makes the total objective value 321, 28 cheaper than the Chain model's objective value. The sailing costs are decreased the most, but also the visiting costs are smaller. The transhipment costs however, are increased, but not enough for the Basic solution to be more profitable.

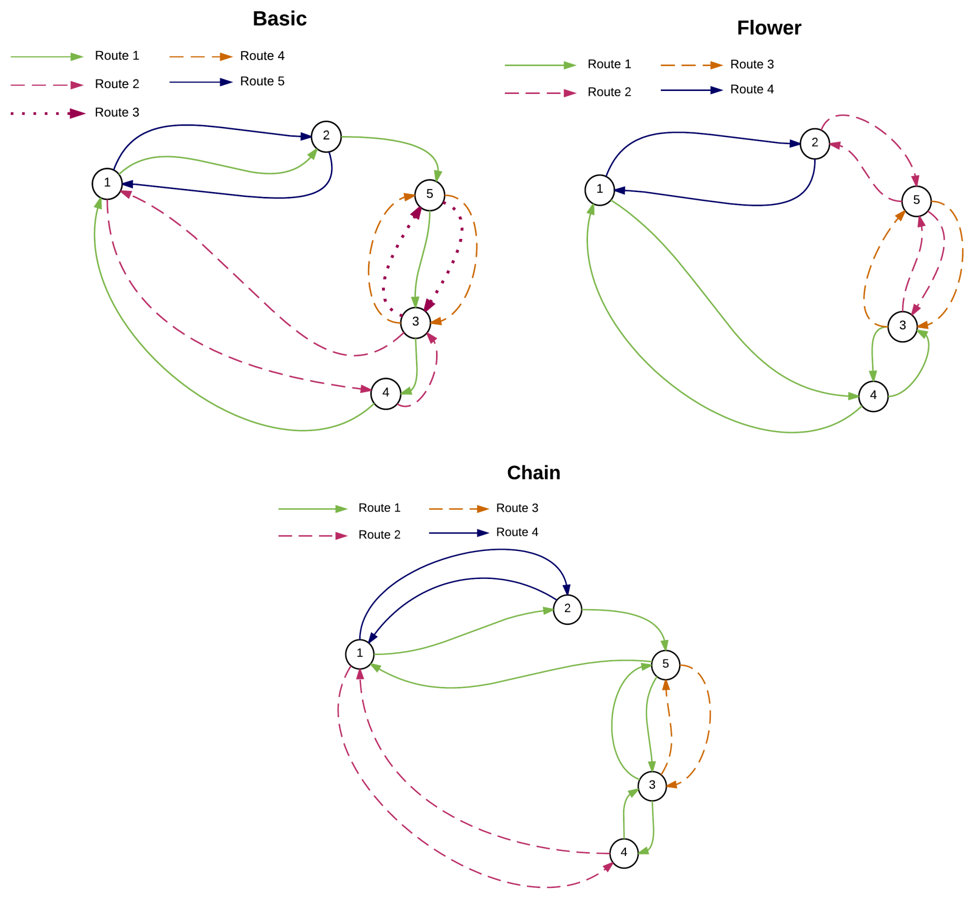


Figure 12.7: Advantages of complex routes. Routes from the different original models generated for data input H[5214].

The Flower model's solution is also possible to create with the Chain model, but the Chain model has generated yet another, and cheaper, solution. This solution includes a Chain route which is not allowed by the Flower model, thus it has more than one port that are visited twice. Both port 5 and Port 3 are visited twice by Route 1, creating a "Chain-shaped"-route. This solution have the same sailing costs as the Flower model's, a little higher visiting costs, but Route 1 decreases the solution's transhipment costs and the total cost ends up at 313, which is 8 less than the Flower solution and 36 less than the Basic solution. This example clearly shows the advantages and saving opportunities of allowing for more complex routes like the Flower model and Chain model do.

The gain of internal transhipment

For data instance G[519], the optimal solutions are different for each of the three models. The Flower model obtains the best optimal solution, hence this solution contains at least one route with more than two loops. The three models' different solutions are illustrated in Figure 12.8.

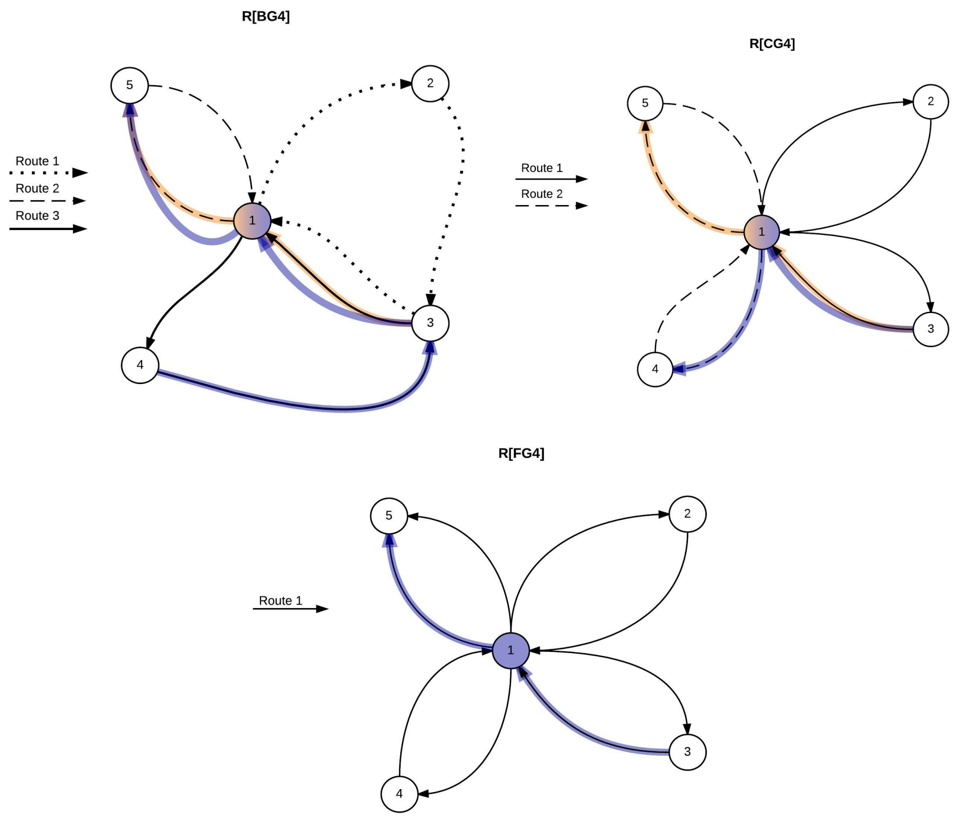


Figure 12.8: The three models give different solutions for data instance G[519]. The networks they make are depicted in the figure. The upper left routes are made by the Basic model, the upper right routes are made by the Chain model and the lower route is made by the Flower model. The blue and orange lines illustrate the transhipment. In the Flower route internal transhipment occurs in the butterhub.

The solution from the Basic model for instance G[519] tranships two different demands in the butterhub (port 1), and as Table 12.5 shows, this transhipment leads to an additional cost of 5. The Chain model's solution of G[519] consists of two butterfly route with two loops each. This solution also tranships two demands in the butterhub, and gets a transhipment cost of 9. This is higher than the Basic model, but the Chain model saves on the sailing cost and visiting cost and ends up with a lower total cost than the Basic model. The best optimal solution, generated by the Flower model for data instance G[519], involves internal transhipment in the butterhub, illustrated by the blue line in Figure 12.8. This transhipment adds a cost of 2 to the objective value, and the Flower model gets an objective value of 32. By studying the output in further depth, it looks like if there are many demands and the ship capacity of a route fills up more due to this, it can pay off to tranship internally in stead of creating a new route with capacity for the demands.

A different Chain route

A Chain route is defined as a route where at least one port is visited twice, but no ports are visited more than twice. This allows for the "Chain-shaped" routes as Route 1 in the Chain model's solution in Figure 12.7. It also allows for more complex looking routes like the one shown in Figure 12.9.

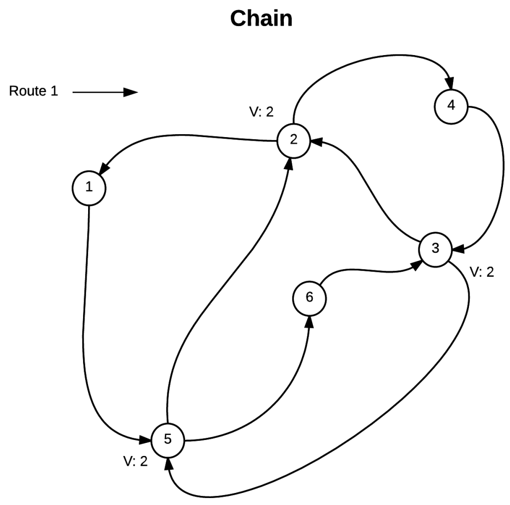


Figure 12.9: A different Chain route. No ports are visited more than twice, but several ports are visited more than once.

The demand of maximum two visits per port is complied, so it clearly qualifies for being a Chain route, but it is interesting that the route structure might be a little different than what one first might think of as a typical Chain route. A lot of the solutions from the testing resulted in the Chain model saving more than the two other models due to route structures similar to the one in Figure 12.9. The savings are mostly caused by the saving in sailing costs because of the rounding up when calculating the sailing costs (this concept is exemplified in detail in Table 12.6), but also the reduction in the visiting costs and the transhipment costs are playing a part in making these routes more profitable.

12.3 Analysis of the transit time models

In this section, the computational results and economical solutions of the transit time models (the Basic TT model, the Flower TT model and the Chain TT model) are analysed. The three transit time models' computational results are compared and discussed in section 12.3.1, and the solutions are compared in Section 12.3.2.

As in the original models' analysis, all the transit time models include the chosen symmetry breaking constraints and valid inequalities from Section 11.2 and Section 11.3. Table 12.1 shows an overview of what is included in the models and the route set size is 9, like stated in 12.1.

12.3.1 Technical results

In this section the technical results from the testing of the transit time models are analysed. It has been run tests on 15 different data instances, where 5 of them are of a General network type, 5 are of a Feeder network type and 5 are of a Hub & Spoke network type. The inputs have either 2 and 3 ship types, a number of demands between 5 and 14, and a number of ports between 5 and 9.

General technical characteristics

Table 12.7 shows the run times, the best bound and the gap of the transit time models.

As for the original modes, the tendency is clear. Chain TT has the longest run times, followed by Flower TT. Basic TT has significantly lower run times than the other two.

Although the Chain TT model does not manage to solve to optimality for several inputs, there are examples where the best solution of the Chain TT model is better than the Flower TT model and the Basic TT model, even though it might not be the optimal solution of the Chain TT model. This is the case for data input T-H[727], where both the Basic TT model and the Flower TT model have solved to optimality with an objective value of 346 and 334, respectively. The Chain TT model however, has not managed to prove its solution optimal, but the best integer solution of 312 is still better than both of the other models' solutions.

Table 12.8 shows the number of nodes for the run times that solved to optimality within the time limit. The Chain TT model has the most nodes for all the test instances, and with only one exception, the Basic TT model has the fewest.

	BT			FT			CT		
	R.t.	**Best int. sol.**	**Gap**	**R.t.**	**Best int. sol.**	**Gap**	**R.t.**	**Best int. sol.**	**Gap**
T-G[526]	44.803	78	0 %	276.767	78	0 %	270.562	76	0 %
T-G[535]	267.153	65	0 %	551.891	65	0 %	1288.56	65	0 %
T-G[627]	435.009	113	0 %	2634.67	113	0 %	-	*114*	8 %
T-G[728]	2325.57	100	0 %	-	*120*	35 %	-	*	*
T-G[9311]	-	*	*	-	*	*	-	*	*
T-F[525]	80.805	72	0 %	333.358	70	0 %	1074.52	72	0 %
T-F[529]	145.237	78	0 %	1037.14	78	0 %	5970.06	78	0 %
T-F[727]	3004.09	99	0 %	11949.1	99	0 %	-	*115*	41 %
T-F[5211]	145.963	90	0 %	1224.75	90	0 %	13605.8	90	0 %
T-F[7211]	19591.5	137	0 %	-	*172*	54 %	-	*	*
T-H[525]	85.225	252	0 %	214.373	252	0 %	511.394	252	0 %
T-H[529]	67.099	189	0 %	319.862	189	0 %	2112.77	186	0 %
T-H[727]	7397.42	192	0 %	-	*228*	28 %	-	*211*	22 %
T-H[5214]	1515.05	346	0 %	3818.98	334	0 %	-	*312*	5 %
T-H[7211]	-	*225*	28 %	-	*241*	50 %	-	*258*	66 %

Table 12.7: Run time, best integer solution and gap from testing of the transit time models. * = No results due to no found integer feasible solution. Italic numbers are when the result is not proven optimal but integer feasible. - = Run time limit reached.

	BT	**FT**	**CT**
T-G[526]	709	5697	6685
T-G[535]	851	2131	12515
T-F[525]	825	5975	27647
T-F[529]	6561	5267	49577
T-F[5211]	2327	11761	90388
T-H[525]	437	1137	7881
T-H[529]	2189	2953	35349

Table 12.8: Number of nodes from testing of the transit time models

Impact of the order split size

A difference between the transit time models and the original models is that the orders in the transit time models are unsplitable. Thus, the original models are less restricted and allow more flexibility with regards to the demand flow than the transit time models. However, in the transit time model, the order split size (OSS) has to be set. This is explained in depth in Section 8.2. Below we discuss three aspects of the order split size that are especially interesting.

First; the run times clearly decrease when the order split size is enlarged. This is illustrated in Table 12.9, which shows the relative run time sizes for each model for data inputs with different order split sizes (1, 2 and 9).

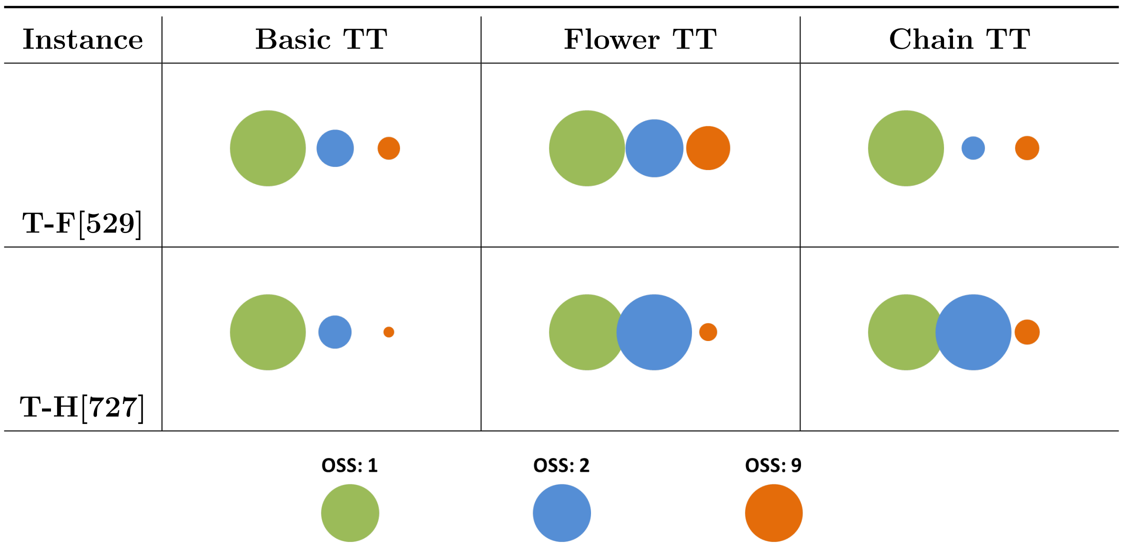


Table 12.9: Relative run times with different order split sizes (OSS) from testing of the transit time models

The second is that if the OSS is set to 1 and the maximum transit times are very high, the solution space is practically the same in both the original models and the transit time models. Although the flow variables in theory can be fractional, we have observed that throughout all testing, the solutions always have integer flow-variables. The disadvantage of setting the OSS=1 is that there will be created many more variables and constraints, which probably will increase the run time. In other words, there is a trade off between the economical gain of allowing solutions that need the order split sizes to be small, and the run time gain of larger order split sizes, i.e. forcing more containers to have the same path.

The third interesting observation is that for the Chain TT model and the Flower TT model on data instance T-H[727], neither managed to solve to optimality within the run time limit when the order split size were 1 or 2. However, when it was set to 9, the run times are much less. For the OSS=9 case, the Flower TT model found an optimal objective value of 198. For the corresponding data input in the *original* Flower model, the model did not manage to solve to optimality within the run time limit, and the best found integer solution's objective value is 320. In other words, the run time gain of forcing the OSS to be 9 made the Flower TT model able to find an integer feasible solution within the run time limit that were better than the not optimal but integer feasible solution the original Flower model found within the same run time limit. Thus, there can be a run time gain of using the transit time models to force the order split, which for an input where the run time reaches a run time limit results in an economic gain as well.

12.3.2 Comparison of solutions

In this section, the solutions from transit time models are explored and presented. Beyond the tests done for the technical analysis, there are run a some more tests on additional data inputs in order to explore different properties of the model and prove the different model features.

The impact of the urgency level

An interesting aspect to explore is how the models including transit time respond to different urgency levels. The urgency levels are presented in Table 10.5. The total costs of the urgency level test results are shown in Table 12.10. Some especially interesting results are the solutions the models found for data input T-F[5211], with the three different urgency levels.

As Table 12.10 shows, the Chain TT model did not solve to optimality within the time limit for T-H[5214] when the urgency level is *Moderate* and *Uncritical*, but for *Uncritical*, the best integer solutions' objective value is better than in the other models' solutions.

For data instance T-F[5211], the models find equal solutions when the urgency level is *Uncritical*.

		BT	**FT**	**CT**
T-G[524]	*U*	54	54	53
	M	49	49	49
	UC	49	49	49
T-H[5214]	*U*	*	*	*
	M	425	415	*432*
	UC	346	334	*312*
T-F[5211]	*U*	116	114	97
	M	100	100	91
	UC	90	90	90

Table 12.10: Total costs of the three models' solutions' with different urgency levels (U(Urgent), M(Moderate) and UC(Uncritical)). Italic numbers are when the result is not proven optimal but integer feasible. * = No results due to no found integer feasible solution.

These solutions are not are not fulfilling the stricter maximum transit time demands when the urgency level becomes *Moderate*, and the models have to find new solutions. Within the new solution space, Basic TT and Flower TT find solutions that are more expensive than Chain TT's solution. Thus, Chain TT has found a route network that comply with the stricter maximum transit times that neither Flower TT or Basic TT allow. For the highest urgency level, *Urgent*, Flower TT and Chain TT each find solutions that are better than Basic TT's solution. Figure 12.10 shows the route structures of these solutions.

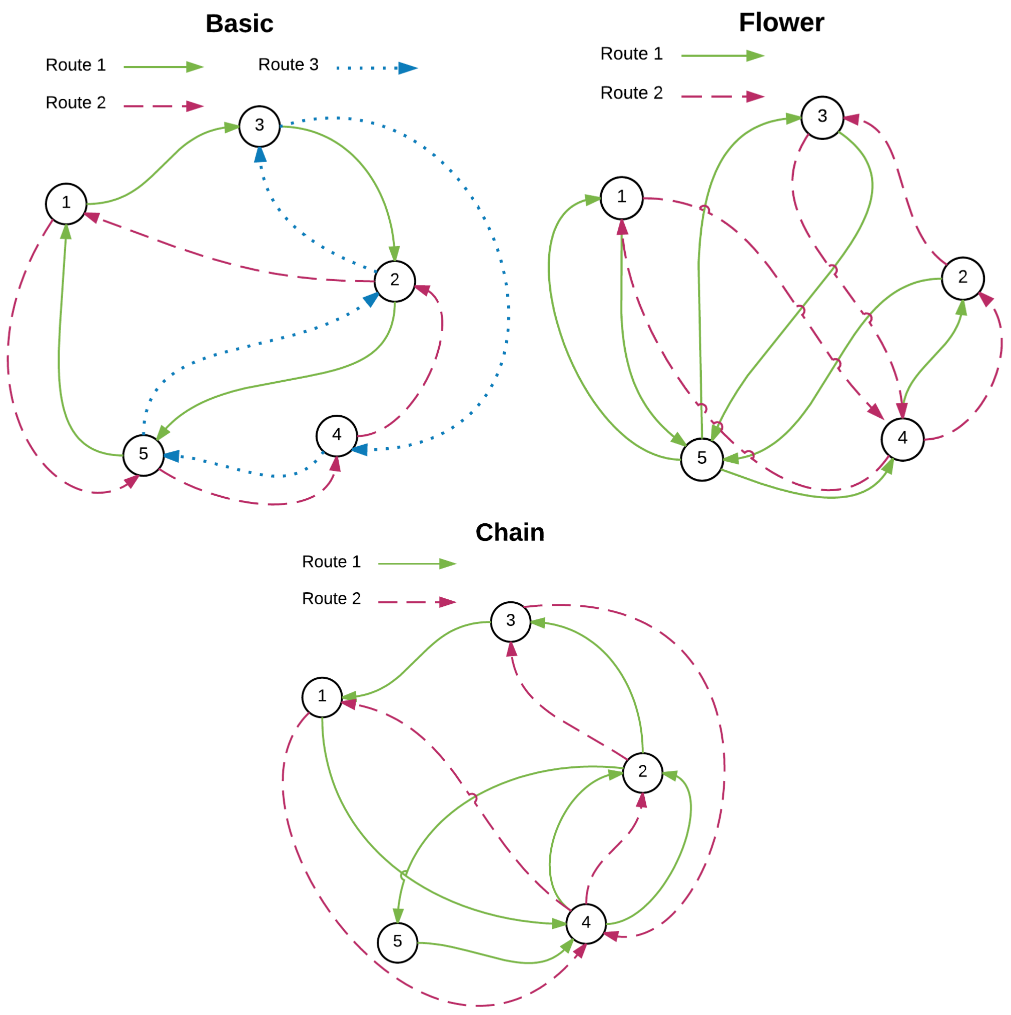


Figure 12.10: The three models give different solutions for data instance T-F[5211] with an urgency level of *Urgent*. The networks they make are depicted in the figure. The Basic model's solution of 116 is the most expensive one, then comes the Flower model's solution of 114, and the Chain model's solution of 97 is the best one.

The Basic model generates three simple routes, while the Flower model generates one three-looped Flower route and one butterfly route. The Chain model's solution, which is the best one, generates a Chain route where both port 2 and port 4 is visited twice, in addition to the same butterfly route as the Flower model.

These results can indicate that the profitability of using the Flower model or the Chain model increases when the transit time urgency level is higher. The faster we need a demand to travel from the origin port to the destination port, the more there is to save on the ability to generate complex routes, most likely due to the saved time on transhipment between routes.

The effect of including transit time

The ability to control the maximum transit times of the orders is the main difference between the original models and the transit time models. In order to explore the effect on the solution of the added transit time constraints, the solution from the Flower model for data input F[5211] have been compared to the transit times from the corresponding transit time data input T-F[5211]. These two inputs have the exact same amount of demand/orders between all OD-pairs. The difference is that for T-F[5211], the demands are split into orders with a given maximum transit time. In this example, the urgency level is "Urgent". Thus, taking the maximum transit times from T-F[5211] and comparing these to the actual transit times from the demand paths generated in the F[5211]-solution shoes which of the orders from the T-F[5211] that would have been delayed with the solution from the original Flower model. Table 12.11 shows the orders with the maximum transit time from the Flower TT model, the actual transit time from the original Flower model and whether or not each order would have been delayed.

Table 12.11 shows that seven of the orders would have been delayed in the Flower model solution with respect to the urgent transit time demands. Although the data inputs, urgency levels and transit times in the tests are fictional, this illustrates that the solutions from the original models might not comply with the transit time demands. The option of setting such a transit time limit can be useful.

Order (o,d,ω)	**TT max**	**Actual TT in the original model**	**Delivered on time**
(1,2,1)	0.80	1.17	✗
(2,1,1)	1.00	0.78	✓
(2,3,1)	0.29	0.23	✓
(2,3,2)	0.33	0.23	✓
(2,5,1)	1.10	0.61	✓
(2,5,2)	0.77	0.61	✓
(2,5,3)	0.88	0.61	✓
(2,5,4)	1.10	0.61	✓
(3,4,1)	0.50	1.24	✗
(3,5,1)	1.14	1.11	✓
(3,5,2)	1.42	1.11	✓
(4,2,1)	0.34	0.63	✗
(4,2,2)	0.39	0.49	✗
(4,2,3)	0.49	0.49	✗
(4,2,4)	0.34	0.49	✗
(5,1,1)	0.80	1.40	✗
(5,2,1)	1.10	0.61	✓
(5,2,2)	0.77	0.61	✓
(5,3,1)	1.14	0.80	✓
(5,3,2)	1.42	0.80	✓
(5,3,3)	1.00	0.80	✓
(5,3,4)	1.14	0.84	✓
(5,3,5)	1.42	0.84	✓
(5,4,1)	0.51	0.40	✓

Table 12.11: Comparison of the actual transit times from the original model and the maximum transit times from the transit time models

Chapter 13

Concluding Remarks

The work done in this report is motivated by the conviction that there exists a huge potential for operations research in the liner shipping industry.

Three extensive mathematical models for the liner shipping network design problem (LS-NDP) have been developed. The three models have been extended to also take into account transit time, making it six formulations in total. Besides whether they incorporate transit time, the six models differ in which route structures they enable.

We take the perspective of a liner shipping company. The objective of all the models is to minimize costs by designing an efficient network of routes that satisfy all demand. All the formulations include important features as transhipment, transhipment cost, a weekly frequency requirement and a heterogeneous fleet vessel.

13.1 Summary and conclusion

One of the key focus areas has been on developing formulations that allows for more flexible route structures. This is motivated by knowing that a significant number of the routes sailed by the global liners today would not have been possible to make with the existing models for the LS-NDP. The first formulation, called *the Basic model*, only allows simple route networks. The two other models present different ways of incorporating complex route structures. *The Flower model* allows the routes in the network to visit one port in the route multiple times, while *the Chain model* incorporates the possibility of a route visiting several ports in the route twice. There has not been made any compromises in allowing the complex route structures, and all transhipment costs are calculated correctly.

The three models have been extended to incorporate transit time. These models make it possible to specify a maximum transit time for each container, and they also allow for different transit times between the same pair of origin and destination ports.

The complexity of the models is severe, and they also suffer from symmetry issues. Motivated by this, there has been made a strong effort in order to strengthen the formulations. A number of symmetry breaking constraints (SBCs) and problem specific valid inequalities are proposed, and their effect is thoroughly investigated. Several of the strengthening formulations show good effect and are included in the further computational study.

Technical results show that the Basic model performs substantially better than the models allowing complex route structures, in terms of run time. In particular, the Chain model spends a lot of time proving optimality. This finding was not surprising considering that the Flower and Chain models are based on the formulation of the Basic model, and contain a significantly larger amount of variables and constraints.

The economical analysis shows that allowing more complex route structures pays off, because this leads to a better allocation of vessels. Of the instances tested, the Chain model most often has the best integer solution at cut off time. This is a mitigating factor, considering that Chain is the by far slowest of the model proving optimality, it is quite fast finding good integer solutions.

Results from the computational study of the transit time models showed that, as expected, stricter transit times in general lead to more expensive route networks. However, the most interesting finding was how the models' solutions responded differently to more rigid levels of urgency. It appears that the Basic transit time model is more sensitive to stricter transit time requirements than the transit time models allowing complex route structures. The advantage of allowing complex route structure is thus amplified when also considering transit time.

13.2 Further research

For further research we find the repositioning of empty containers especially interesting, and relevant in light of the work done in this report. The repositioning of empty containers is a great challenge because of the huge global trade imbalances in the world leads to systematic accumulation of empty containers in some regions, while other regions that exports more than it imports will face a shortage of containers. Taking into account the repositioning of empty containers may alter the optimal route network for the company. This, in combination with the fact that repositioning of empty containers being a steadily bigger issue, make this seem like an interesting direction to investigate for further research.

In this paper several strengthening formulations are presented, and many of them show good effect. However, we still think there is a great potential of enhancing the computational efficiency of the models. Especially in the transit time models we think there can be much to be gained from preprocessing of variables.

Bibliography

R. Agarwal and Ö. Ergun. "Ship scheduling and network design for cargo routing in liner shipping". *Transportation Science*, 42:175–196, 2008.

R. Agarwal and Ö. Ergun. "Network design and allocation mechanisms for carrier alliances in liner shipping". *Operations Research*, 2010.

J. Alvarez. "Joint routing and deployment of a fleet of container vessels". *Maritime Econom. Logist.*, pages 186–208, 2009.

American Association of Port Authorities. "World port rankings 2012 ", 2013. URL `http://aapa.files.cms-plus.com/statistics/world%20port%20rankings%202012.pdf`.

R. Asariotis, H. Benamara, H. Finkenbrink, J. Hoffmann, A. Jaimurzina, A. Premti, V. Valentine, and F. Youssef. "UNCTAD: Review of Maritime Transport 2014". *Review of Maritime Transport*, 2014.

C. Barnhart, N.L. Boland, L.W. Clarke, E.L. Johnson, G.L. Nemhauser, and R.G. Shenoi. "Flight String Models for Aircraft Fleeting and Routing". *Transportation Science*, 32(3): 208–220, 1998.

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

T. B. Boffey, E. D. Edmond, A. I. Hinxman, and C. J. Pursglove. "Two Approaches to Scheduling Container Ships with an Application to the North Atlantic Route". *The Journal of the Operational Research Society*, 30(5):413–425, 1979.

Berit D. Brouer, J. Fernando Alvarez, Christian E. M. Plum, David Pisinger, and Mikkel M. Sigurd. "A Base Integer Programming Model and Benchmark Suite for Liner-Shipping Network Design". *Transportation Science*, 48(2):281–312, 2014. doi: 10.1287/trsc.2013.0471. URL `http://dx.doi.org/10.1287/trsc.2013.0471`.

P. Cariou. "Is slow steaming a sustainable mean for reducing liner shipping CO2 emissions?

", 2014. URL http://old.mareforum.com/euromed_presentations_2010/cariou_paper.pdf.

R. Cheung and C. Chen. "A Two-Stage Stochastic Network Model and Solution Methods for the Dynamic Empty Container Allocation Problem". *Transportation Science*, 32(2):142–162, 1998.

S.C. Cho and A.N. Perakis. "Optimal liner fleet routing strategies". *Maritime Policy & Management*, pages 249–259, 1996.

M. Christiansen and B. Nygreen. "A method for solving ship routing problems with inventory constraints". *Annals of Operations Research*, 81, 1998.

M. Christiansen, K. Fagerholt, and D. Ronen. "Ship Routing and Scheduling: Status and Perspectives". *Transportation Science*, 38(1):1–18, 2004.

T. Chuang, C. Lin, J. Kung, and M. Lin. "Planning the route of container ships: A fuzzy genetic approach". *Expert Systems with Applications*, 37(4):2948–2956, 2010.

D. Cieslik. "Network design problems Network Design Problems". In CA. Floudas and PM. Pardalos, editors, *"Encyclopedia of Optimization"*, pages 2539–2545. Springer US, 2009.

CNSS. "Slow steaming to stay ", 2013. URL http://cleantech.cnss.no/best-practices/slow-steaming-to-stay.

K. Fagerholt. "Optimal fleet design in a ship routing problem". *Internat. Transactions Oper. Res.*, pages pp. 453–464, 1999.

K. Fagerholt. "Designing optimal routes in a liner shipping problem". *Maritime Policy Management*, pages 259–268, 2004.

K. Fagerholt and H. Lindstad. "Optimal policies for maintaining a supply service in the Norwegian Sea". *Omega*, 28(3):269–275, 2000.

K. Fagerholt, T.A.V. Johnsen, and H Lindstad. "Fleet deployment in liner shipping: a case study". *Maritime Policy & Management*, 36(5):397–409, 2009.

T. D. Heaver and D. Uyeno. "Planning and scheduling for efficiency in liner shipping". *Maritime Policy & Management*, 14:109–125, 1987.

A. Imai, K. Shintani, and S. Papadimitriou. "Multi-port vs. Hub-and-Spoke port calls by containerships". *Transportation Research Part E: Logistics and Transportation Review*, 45 (5):740 – 757, 2009.

IHS Global Insight. "Valuation of the liner shipping industry - Economic contribution and Liner

Industry Operations", 2009. URL http://www.worldshipping.org/pdf/Liner_Industry_Valuation_Study.pdf.

A.N. Jaramillo and D.I. Perakis. "Fleet deployment optimization for liner shipping part 2. Implementation and results.". *Maritime Policy Management*, pages 235–262, 1991.

D. S. Johnson, J. K. Lenstra, and A. H. G. Rinnooy Kan. "The complexity of the network design problem". *Networks*, 8(4):279–285, 1978.

M.G. Karlaftis, K. Kepaptsoglou, and E. Sambracos. "Containership routing with time deadlines and simultaneous deliveries and pick-ups ". *Transportation Research Part E: Logistics and Transportation Review*, 45(1):210 – 221, 2009.

C.V. Karsten, D. Pisinger, S. Ropke, and B.D Brouer. The time constrained multi-commodity network flow problem and its application to liner shipping network design. *Transportation Research Part E: Logistics and Transportation Review*, 76(0):122 – 138, 2015.

K.H. Kjeldsen. "Classification of Ship Routing and Scheduling Problems in Liner Shipping". *INFOR: Information Systems and Operational Research*, 49(2):139–152, 2011.

B. Korte and J. Vygen. Network design problems. In *"Combinatorial Optimization"*, volume 21 of *Algorithms and Combinatorics 21*, pages 467–500. Springer Berlin Heidelberg, 2006. doi: 10.1007/3-540-29297-7_20.

F. Kydland. "Simulation of Liner Operations". 1969.

D.E. Lane, T.D. Heaver, and D. Uyeno. "Planning and scheduling for efficiency in liner shipping". *Maritime Policy Management*, 1987.

S.A. Lawrence. *"International sea transport: the years ahead"*. Lexington Books Lexington, Mass, 1972.

Maersk. Ae10-roundtrip, 2014a. URL http://www.maerskline.com/~/media/maersk-line/Countries/int/Routenet/pdfs/asia-europe/ae10-roundtrip.pdf.

Maersk. "Containers and containerization ", 2014b. URL http://www.maersk.com/en/industries/transport#containers.

Q. Meng and T. Wang. "A scenario-based dynamic programming model for multi-period liner ship fleet planning". *Transportation Research Part E: Logistics and Transportation Review*, 47(4):401–413, July 2011.

Q. Meng, T. Wang, and S. Wang. "Short-term liner ship fleet planning with container transshipment and uncertain container shipment demand". *Eur. J. Oper. Res.*, 223(1):96–105, 2012.

Q. Meng, S. Wang, H. Andersson, and K. Thun. "Containership Routing and Scheduling in Liner Shipping: Overview and Future Research Directions". *Transportation Science*, 48(2): 265–280, 2014.

M. C. Moura, M. V. Pato, and A. C. Paixa. "Ship assignment with hub and spoke constraints". *Maritime Policy & Management*, 29(2):135–150, 2002.

T.E. Notteboom. "The Time Factor in Liner Shipping Services". *Maritime Economics and Logistics*, 8(1):19–39, 2006.

T.E. Notteboom and J. Rodrigue. "Containerisation, box logistics and global supply chains: The integration of ports and liner shipping networks". *Maritime Economics & Logistics*, 10 (1):152–174, 2008.

B. Nygreen, M. Christiansen, K. Haugen, T. Bjørkvoll, and Ø. Kristiansen. "Modeling Norwegian petroleum production and transportation". *Annals of Operations Research*, 82(0): 251–268, 1998. ISSN 0254-5330.

C.A. Olson, E.E. Sorenson, and W.J. Sullivan. "Medium-range scheduling for a freighter fleet". *Operations research*, 17:565–582, 1969.

A.N. Perakis and D.I. Jaramillo. "Fleet deployment optimization for liner shipping part 1. Background, problem formulation and solution approaches". *Maritime Policy Management*, pages 183–200, 1991.

R. Pesenti. "Hierarchical resource planning for shipping companies ". *European Journal of Operational Research*, 86(1):91 – 102, 1995.

B.J. Powell and D.I. Perakis. "Fleet deployment optimization for liner shipping: An integer programming model". *Maritime Policy & Management*, 24(2):183–192, 1997.

H.N. Psaraftis. "Foreword to the focused issue on maritime transportation". *Transportation Science*, 1999.

H.N. Psaraftis and C.A. Kontovas. “Ship Emissions, Costs and their Tradeoffs”. *Advances in Maritime Logistics and Supply Chain Systems*, 2010.

K. Rana and R. G. Vickson. "Routing Container Ships Using Lagrangean Relaxation and Decomposition". *Transportation Science*, 25(3), 1991.

L.B. Reinhardt and D. Pisinger. "A branch and cut algorithm for the container shipping network design problem". *Flexible Services Manufacturing Journal*, pages 349–374, 2011.

L.B. Reinhardt, B. Kallehauge, A. Nørrelund, and A. Olsen. "Network design models for container shipping". Technical report, Technical report, Centre for Traffic and Transport, Technical University of Denmark, 2007.

D. Rönen. "Cargo ships routing and scheduling: Survey of models and problems ". *European Journal of Operational Research*, 12(2):119 – 126, 1983.

E. Sambracos, J.A. Paravantis, C.D. Tarantilis, and C.T. Kiranoudis. "Dispatching of small containers via coastal freight liners: The case of the Aegean Sea". *European Journal of Operational Research*, pages 365–381, 2004.

V. Schmid. "Hybrid large neighborhood search for the bus rapid transit route design problem". *European Journal of Operational Research*, 238(2):427–437, 2014.

K. Shintani, A. Imai, E. Nishimura, and S. Papadimitriou. "The container shipping network design problem with empty container repositioning". *Transportation Research Part E: Logistics and Transportation Review*, 43(1):39 – 59, 2007. ISSN 1366-5545.

M.M. Sigurd, N.L. Ulstein, B. Nygreen, and D.M. Ryan. "Ship Scheduling with Recurring Visits and Visit Separation Requirements". In *Column Generation*, pages 225–245. Springer US, 2007.

D.P. Song and P.M. Panayides. "A conceptual application of cooperative game theory to liner shipping strategic alliances". *Maritime Policy Management*, 2002.

S. Ting and G. Tzeng. "Ship scheduling and service network integration for liner shipping companies and strategic alliances". *Journal of the Eastern Asia Society for Transportation Studies*, 5, 2003.

J. Vesterager. "Alphaliner: Verdens havne er ikke klar til gigantskibe". *Shippingwatch*, 2012. URL `http://shippingwatch.dk/Havne/article4943407.ece`.

T. Walsh. "symmetry breaking constraints: Recent results ". 2012.

S. Wang, Q. Meng, and Z. Sun. "Container routing in liner shipping". *Transportation Research Part E: Logistics and Transportation Review*, 49(1):1 – 7, 2013.

L.A. Wolsey. *"Integer Programming"*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998. ISBN 9780471283669. URL `http://books.google.no/books?id=x7RvQgAACAAJ`.

World Shipping Council. "A coordinated voice for the liner shipping industry ", 2014a. URL `http://www.worldshipping.org/`.

World Shipping Council. "A coordinated voice for the liner shipping industry", 2014b. URL `http://www.worldshipping.org/about-the-industry/global-trade/trade-routes`.

S. Yan, C. Chen, and S. Lin. "Ship scheduling and container shipment planning for liners in short-term operations". *Journal of marine science and technology*, 14(4):417–435, 2009.

Appendix A

Complete mathematical models

A.1 Variables, parameters and sets

Variables in alphabetical order

a_{kr}	- The number of vessels of ship type k sailing route r
b_{ik}	- The number of times port i is visited by ship type k each week
c_{ijodkr}	- 1 if the net flow in port i of demand (o,d) associated with route r is non-positive, 0 otherwise. *(Original models)*
$c_{ijod\omega kr}$	- 1 if the net flow in port i of order (o,d, ω) associated with ship type k and route r is non-positive, 0 otherwise. *(TT models)*
d_{ir}	- 1 if port i is set to depot in route r, 0 otherwise.
e^F_{ir}/e^L_{ir}	- 1 if port i is in the first/last loop (n_{ir}=1/n_{ir}=l_r), 0 otherwise
f_{ijodkr}	- Flow of the demand (o,d) transported by a vessel of type k on arc (i,j) in route r *(Original models)*
$f_{ijod\omega kr}$	- 1 if order (o,d,ω) is transported by a vessel of type k on arc (i,j) in route r, 0 otherwise *(TT models)*
g^N_{ijr}/g^S_{ijr}	- 1 if port i's loop/order number is greater than port j's, 0 otherwise
h_{ijr}	- 1 if port i and port j are in the same loop ($n_{ir} = n_{jr}$), 0 otherwise
l_r	- The number of loops in route r
m_{ir}	- 1 if port i has multiple arcs (i.e. a butterhub) in route r, 0 otherwise
n_{ir}	- The loop number of a port i in route r
o_{ijr}	- Difference between port i and port j's loop number max$\{n_{ir}$-n_{jr},0$\}$

p^N_{ijr}	- 1 if i and j are in neighbouring loops and port i's loop is sailed before port j's loop
p^R_{ijr}	- 1 if both port i and port j are visited in route r, 0 otherwise.
p^S_{ijr}	- 1 if port i is in the last loop (e^L_{ir}=1) and port j is in the first loop (e^F_{jr}=1)
q_{imodr}	- The positive pairwise difference between out-flow and in-flow of demand (o, d) in neighbouring loops in butterfly route r *(Original models)*
$q_{imod\omega r}$	- The positive pairwise difference between out-flow and in-flow of order (o, d, ω) in neighbouring loops in butterfly route r *(TT models)*
r_{ir}	- 1 if port i is visited in route r, 0 otherwise.
s_{ir}	- Artificial variable that represents a port i's order in route r, counted from the depot.
t_{iodr}	- The amount of demand (o,d) that is transhipped in port i in route r *(Original models)*
$t_{iod\omega r}$	- The amount of demand (o,d,ω) is transhipped to route r in port i *(TT models)*
t^I_{iodr}	- Internal transhipment of demand (o, d) in butterhub i of butterfly route r, *(Original models)*
$t^I_{iod\omega r}$	- Internal transhipment of order (o, d, ω) in butterhub i of butterfly route r *(TT models)*
t^E_{iodr}	- External transhipment of demand (o, d) in port i of route r *(Original models)*
$t^E_{iod\omega r}$	- External transhipment of order (o, d, ω) in port i of route r *(TT models)*
x_{ijkr}	- 1 if a vessel of type k is used on arc (i,j) by route r, 0 otherwise
y_{kr}	- 1 if a vessel of type k is used on route r, 0 otherwise

Parameters in alphabetical order

C^P_{ik}	- Visiting cost (per visit) at port i for ship type k	$i \in \mathcal{P}, k \in \mathcal{K}$
C^S_k	- Sailing cost per week for ship type k	$k \in \mathcal{K}$
C^T_i	- Transhipment cost per unit of demand (o, d) port i	$i \in \mathcal{P}$
D_{od}	- Weekly demand from port o to port d	$(o, d) \in \mathcal{D}$
$D^B_{od\omega}$	- 1 if there is demand of order ω from port o to port d, 0 otherwise	$(o, d, \omega) \in \mathcal{O}$
$D^Q_{od\omega}$	- Weekly quantity (in TEU) of order (o,d,ω)	$(o, d, \omega) \in \mathcal{O}$
L	- Max route length (NM)	
M_k	- Number of vessels of ship type k in fleet	$k \in \mathcal{K}$
N	- Number of ports	
S_k	- Capacity of ship type k	$k \in \mathcal{K}$
T_{ij}	- Distance (in NM) on arc (i,j)	(i,j) $\in \mathcal{A}$
T_{ijk}	$= T_{ij}/V_k$. Sailing time on arc (i,j) for ship type k	(i,j) $\in \mathcal{A}, k \in \mathcal{K}$
$T^T_{od\omega}$	- Transit time requirement (in weeks) for order (o,d,ω)	$(o, d, \omega) \in \mathcal{O}$
V_k	- Velocity (NM/Weeks) of ship type k	$k \in \mathcal{K}$
W	- Max route time (Weeks)	

Sets in alphabetical order

$\mathcal{A}$	- Arcs, (i,j)	$i, j \in \mathcal{P}$
$\mathcal{D}$	- Demands, (o,d)	$o, d \in \mathcal{P}$
$\mathcal{K}$	- Ship types, k	
$\mathcal{N}$	- Nodes, i,j	$i, j \in \{1..2N\}$
Ω	- The set of possible order indices per O-D pair,	ω
$\mathcal{O}$	- Orders, (o,d,ω)	$o, d \in \mathcal{P}, \omega \in \Omega$
$\mathcal{P}$	- Ports, i,j	
$\mathcal{R}$	- Routes, r	$r \in \{1.. \sum\limits_{k \in \mathcal{K}} M_k\}, k \in \mathcal{K}$

A.2 Basic model

Sets and indices

$\mathcal{P}$	- Ports, i,j	
$\mathcal{A}$	- Arcs, (i,j)	$i, j \in \mathcal{P}$
$\mathcal{D}$	- Demands, (o,d)	$o, d \in \mathcal{P}$
$\mathcal{K}$	- Ship types, k	
$\mathcal{R}$	- Routes, r	$r \in \{1..\sum_{k\in\mathcal{K}} M_k\}, k \in \mathcal{K}$

Parameters

D_{od}	- Weekly demand from port o to port d	$(o,d) \in \mathcal{D}$
C_i^T	- Transhipment cost per unit port i	$i \in \mathcal{P}$
C_{ik}^P	- Visiting cost (per visit) at port i for ship type k	$i \in \mathcal{P}, k \in \mathcal{K}$
C_k^S	- Sailing cost per week for ship type k	$k \in \mathcal{K}$
V_k	- Velocity (NM/Weeks) of ship type k	$k \in \mathcal{K}$
T_{ij}	- Distance (in NM) on arc (i,j)	$(i,j) \in \mathcal{A}$
T_{ijk}	$= T_{ij}/V_k$. Sailing time on arc (i,j) for ship type k	$(i,j) \in \mathcal{A}, k \in \mathcal{K}$
S_k	- Capacity of ship type k	$k \in \mathcal{K}$
M_k	- Number of vessels of ship type k in fleet	$k \in \mathcal{K}$
N	- Number of ports	
L	- Max route length (NM)	
W	- Max route time (Weeks)	

Variables

x_{ijkr}	- 1 if a vessel of type k is used on arc (i,j) by route r, 0 otherwise
y_{kr}	- 1 if a vessel of type k is used on route r, 0 otherwise
f_{ijodkr}	- Flow of the demand (o,d) transported by a vessel of type k on arc (i,j) in route r
t_{iodr}	- The amount of demand (o,d) that is transhipped in port i in route r
a_{kr}	- The number of vessels of ship type k sailing route r
b_{ik}	- The number of times port i is visited by ship type k each week

Objective function

$$minimize \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} C_k^S \cdot \ a_{kr} + \sum_{i\in\mathcal{P}}\sum_{r\in\mathcal{R}} C_i^T \cdot t_{iodr} + \sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}} C_{ik}^P \cdot \ b_{ik} \tag{A.1}$$

Constraints

$$\sum_{k\in\mathcal{K}} y_{kr} \leq 1, \qquad r \in \mathcal{R} \tag{A.2}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} T_{ij} \cdot x_{ijkr} \leq L y_{kr}, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{A.3}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} T_{ijk} \cdot x_{ijkr} \leq W y_{kr}, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{A.4}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}} x_{ijkr} - y_{kr} \geq 0, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{A.5}$$

$$\sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} (f_{ijodkr} - f_{jiodkr}) = 0, \qquad j \in \mathcal{P}, (o,d) \in \mathcal{D}, j \neq o, j \neq d, \tag{A.6}$$

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} f_{ojodkr} = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{A.7}$$

$$\sum_{i\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} f_{idodkr} = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{A.8}$$

$$\sum_{i\in\mathcal{P}} x_{ijkr} - \sum_{i\in\mathcal{P}} x_{jikr} = 0, \qquad j \in \mathcal{P}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.9}$$

$$t_{iodr} \geq \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} (f_{ijodkr} - f_{jiodkr}), \qquad i \in \mathcal{P}, i \neq o, i \neq d, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{A.10}$$

$$t_{iodr} \cdot \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} (f_{ijodkr} - f_{jiodkr} \geq 0), \qquad i \in \mathcal{P}, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{A.11}$$

$$\sum_{r \in \mathcal{R}} a_{kr} \leq N_k, \qquad k \in \mathcal{K} \tag{A.12}$$

$$a_{kr} \geq \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{P}} T_{ijk} \cdot x_{ijkr}, \qquad k \in \mathcal{K}, r \in \mathcal{R} \tag{A.13}$$

$$\sum_{(o,d) \in \mathcal{D}} f_{ijodkr} \leq S_k \cdot x_{ijkr}, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.14}$$

$$b_{jk} = \sum_{i \in \mathcal{P}} \sum_{r \in \mathcal{R}} x_{ijkr}, \qquad j \in \mathcal{P}, k \in \mathcal{K} \tag{A.15}$$

$$\textbf{SEC}, \qquad r \in \mathcal{R} \tag{A.16}$$

Binary, non-negativity and integrality constraints

$$x_{ijkr}, y_{kr} \in \{0,1\}, \qquad i,j \in \mathcal{P}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.17}$$

$$f_{ijodkr}, t_{iodr}, a_{kr}, b_{jk} \geq 0, \qquad i,j \in \mathcal{P}, (o,d) \in \mathcal{D}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.18}$$

$$a_{kr}, b_{jk} \qquad \text{integer}, \qquad j \in \mathcal{P}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.19}$$

A.2.1 Subtour elimination

Additional variables

r_{ir}	- 1 if port i is visited in route r, 0 otherwise.
d_{ir}	- 1 if port i is set to depot in route r, 0 otherwise.
s_{ir}	- Variable that represents port i's sequence number in route r, counted from the depot.

Additional constraints

$$(\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr})\cdot(1-r_{ir})=0, \qquad i\in\mathcal{P}, r\in\mathcal{R} \tag{A.20}$$

$$\sum_{i\in\mathcal{P}} d_{ir}=1, \qquad r\in\mathcal{R} \tag{A.21}$$

$$s_{jr}-s_{ir}+(N-1)\cdot(1-\sum_{k\in\mathcal{K}} x_{ijkr}+d_{jr})\geq 1, \qquad i\in\mathcal{P}, j\in\mathcal{P}, r\in\mathcal{R} \tag{A.22}$$

$$d_{ir}+\frac{1}{N-1}\cdot s_{ir}\leq 1, \qquad i\in\mathcal{P}, r\in\mathcal{R} \tag{A.23}$$

$$s_{jr}-(N-1)(2-\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr}-d_{ir})\leq 1, \qquad i\in\mathcal{P}, j\in\mathcal{P}, r\in\mathcal{R} \tag{A.24}$$

$$r_{ir}, d_{ir}\in\{0,1\}, \qquad i,j\in\mathcal{P}, r\in\mathcal{R} \tag{A.25}$$

$$s_{ir}\geq 0, \qquad \text{integer} \qquad i\in\mathcal{P}, r\in\mathcal{R} \tag{A.26}$$

A.2.2 Linearizations

Constraints (A.11) can be linearized by substituting them by constraints (A.27-A.29):

Introduce the dummy variable c_{ijodkr}:

c_{ijodkr}	- 1 if the net flow in port i of demand (o,d) associated with route r is non-positive, 0 otherwise.

$$\sum_{k\in\mathcal{K}}(f_{ijodkr}-f_{jiodkr})+D_{od}\cdot c_{ijodkr}\geq 0, \qquad i\in\mathcal{P}, r\in\mathcal{R} \tag{A.27}$$

$$D_{od} \cdot (1 - c_{ijodkr}) - t_{iodr} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.28}$$

$$c_{ijodkr} \in \{0, 1\}, \qquad i, j \in \mathcal{P}, (o, d) \in \mathcal{D}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.29}$$

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - r_{ir} \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.30}$$

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - r_{ir} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.31}$$

A.3 Flower model

Below are all modifications and additions needed from the Basic model to obtain the Flower model.

Modified variables

t^I_{iodr}	- Internal transhipment of demand (o, d) in butterhub i of butterfly route r
t^E_{iodr}	- External transhipment of demand (o, d) in port i of route r

Variable t_{iodr} from the Basic model is substituted by $t^I_{iodr} + t^E_{iodr}$.

Modified objective function

$$minimize \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} C^S_k \cdot \ a_{kr} + \sum_{i \in \mathcal{P}} \sum_{r \in \mathcal{R}} C^T_i \cdot (t^E_{iodr} + t^I_{iodr}) + \sum_{i \in \mathcal{P}} \sum_{k \in \mathcal{K}} C^P_{ik} \cdot \ b_{ik} \qquad \text{(A.32)}$$

Modified constraints

$$\sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} x_{ijkr} - 2(N-1) \cdot r_{ir} \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \qquad \text{(A.33)}$$

Constraints replace constraints A.20 from the Basic model.

Additional variables

m_{ir}	- 1 if port i has multiple arcs (i.e. is a butterhub) in route r, 0 otherwise
n_{ir}	- The loop number of port i in route r
l_r	- The number of loops in route r
e^F_{ir}/e^L_{ir}	- 1 if port i is in the first/last loop (n_{ir}=1/n_{ir}=l_r), 0 otherwise
g^N_{ijr}/g^S_{ijr}	- 1 if port i's loop/order number is greater than port j's, 0 otherwise
h_{ijr}	- 1 if port i and port j are in the same loop ($n_{ir} = n_{jr}$), 0 otherwise
o_{ijr}	- Difference between port i and port j's loop number (n_{ir}-n_{jr})
p^S_{ijr}	- 1 if port i is in the last loop (e^L_{ir}=1) and port j is in the first loop (e^F_{jr}=1)
p^N_{ijr}	- 1 if port i and j are in neighbouring loops and port i's loop is sailed before port j's loop
q_{imodr}	- The positive pairwise difference between out-flow and in-flow of demand (o, d) in neighbouring loops in butterfly route r

Construction of the additional variables

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr} - (N-1)\cdot m_{ir} \leq 1, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.34}$$

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr} - 2\cdot m_{ir} \geq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.35}$$

$$m_{ir} - d_{ir} \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.36}$$

$$l_r \leq \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} x_{ijkr} + (N-1)(1-d_{ir}), \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.37}$$

$$n_{ir} - (N-1)(1-d_{ir}) \leq 0, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.38}$$

$$n_{ir} - n_{jr} - (N-1)(1 - \sum_{k\in\mathcal{K}}(x_{ijkr} + x_{jikr}) + 2d_{ir} + 2d_{jr}) \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.39}$$

$$|(N-1)(s_{ir} - s_{jr}) + n_{ir} - n_{jr}| \geq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.40}$$

$$s_{ir} + d_{ir} + (1 - y_{kr}) \geq 1, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.41}$$

$$n_{ir} \leq l_r, \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.42}$$

$$n_{ir} - 2(1 - e^F_{ir} - d_{ir}) \geq 0 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.43}$$

$$n_{ir} - (N-1)(1-e^F_{ir}) \leq 1 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.44}$$

$$e^F_{ir} - n_{ir} \leq 0 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.45}$$

$$e^L_{ir} - n_{ir} + l_r \geq 1 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.46}$$

$$n_{ir} - l_r + (N-1)(1-e^L_{ir}) \geq 0 \qquad i \in \mathcal{P}, r \in \mathcal{R} \tag{A.47}$$

$$n_{ir} - n_{jr} - (N-1)g^N_{ijr} \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.48}$$

$$n_{ir} - n_{jr} + (N-1)(1-g^N_{ijr}) \geq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.49}$$

$$s_{ir} - s_{jr} - (N-1)g^S_{ijr} \leq 0 \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.50}$$

$$s_{ir} - s_{jr} + (N-1)(1-g^S_{ijr}) \geq 1 \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.51}$$

$$g^N_{ijr} + g^N_{jir} + h_{ijr} = 1 \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.52}$$

$$o_{ijr} - n_{ir} + n_{jr} + (N-1)(1-g^N_{ijr}) \geq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.53}$$

$$o_{ijr} - n_{ir} + n_{jr} - (N-1)(1-g^N_{ijr}) \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.54}$$

$$o_{ijr} - (N-1)(1 - g^N_{jir}) \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.55}$$

$$o_{ijr} - (N-1)(1 - h_{ijr}) \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.56}$$

$$o_{ijr} + o_{jir} + (N-1)h_{ijr} \geq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.57}$$

$$e^L_{ir} + e^F_{jr} - 2p^S_{ijr} \leq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.58}$$

$$e^L_{ir} + e^F_{jr} - 2p^S_{ijr} \geq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.59}$$

$$p^S_{ijr} - p^N_{ijr} \leq 0, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.60}$$

$$o_{jir} + 2g^N_{ijr} + 2h_{ijr} + p^N_{ijr} + 2d_{ir} + 2d_{jr} \geq 2, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.61}$$

$$o_{ijr} + o_{jir} - (N-1)(1 - p^N_{ijr} + p^S_{ijr}) \leq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.62}$$

$$p^N_{ijr} + h_{ijr} \leq 1, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.63}$$

$$q_{imodr} \geq \sum_{k \in \mathcal{K}} (f_{jmodkr} - f_{ijodkr} - D_{od}(4 - m_{jr} - p^N_{ijr} - x_{ijkr} - x_{jmkr}), \qquad (i,j),(j,m) \in \mathcal{A}, r \in \mathcal{R} \tag{A.64}$$

$$t^I_{jodr} \geq \sum_{i \in \mathcal{P}} \sum_{m \in \mathcal{P}} q_{imodr} - t^E_{jodr}, \qquad j \in \mathcal{P}, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{A.65}$$

Binary, non-negativity and integrality constraints

$$m_{ir}, e^F_{ir}, e^L_{ir}, g^N_{ijr}, g^S_{ijr}, h_{ijr}, p^S_{ijr}, p^N_{ijr} \in \{0,1\}, \qquad i,j \in \mathcal{P}, r \in \mathcal{R} \tag{A.66}$$

$$n_{ir}, o_{ir}, l_r, q_{imodr} \geq 0, \qquad \text{integer} \qquad i,m \in \mathcal{P}, (o,d) \in \mathcal{D}, r \in \mathcal{R} \tag{A.67}$$

A.4 Chain model

Below are all modifications and additions needed from the Basic model to obtain the Chain model.

Additions and changes in sets and indices

$$\mathcal{N} \quad \text{- Nodes, } i,j \qquad i,j \in \{1..2N\}$$

Additions and changes in parameters

- The distance between port i and its twin port $i+N$ is 0, i.e. $T_{i,i+N} = T_{i+N,i} = 0$.
- The distance between twin port $i+N$ and twin port $j+N$ is equal to the distance between port i and j, i.e. $T_{i+N,j+N} = T_{ij}$.
- The transhipment cost of twin port $i+N$ is equal to the transhipment cost of port i, i.e. $C^T_{i+N} = C^T_i$. The same applies for the visiting cost.

Modified Constraints

$$b_{ik} = \sum_{j\in\mathcal{P}, j\neq i+N,} \sum_{r\in\mathcal{R}} x_{ijkr}, \qquad i \in \mathcal{P}, k \in \mathcal{K} \tag{A.68}$$

$$\sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} (f_{ijodkr} + f_{i+N,jodkr} - f_{jiodkr} - f_{j,i+N,odkr}) = 0, \\ j \in \mathcal{N}, (o,d) \in \mathcal{D}, j \neq o, j \neq d, j \leq N \tag{A.69}$$

$$\sum_{j\in\mathcal{P}, j\neq o, j\neq o+N} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} (f_{ojodkr} + f_{o+N,jodkr}) = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{A.70}$$

$$\sum_{i\in\mathcal{P}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}} (f_{idodkr} + f_{i,d+N,odkr}) = D_{od}, \qquad (o,d) \in \mathcal{D}, \tag{A.71}$$

A.5 Basic transit time model

Below are all modifications and additions needed from the Basic model to obtain the Basic transit time model.

Additions and changes in sets and indices

Ω	- The set of possible order indices per O-D pair,	ω
$\mathcal{O}$	- Orders, (o,d,ω)	$o, d \in \mathcal{P}, \omega \in \Omega$

Additions and changes in parameters

$D^B_{od\omega}$	- 1 if there is demand of order ω from port o to port d, 0 otherwise	$(o, d, \omega) \in \mathcal{O}$
$D^Q_{od\omega}$	- Weekly quantity (in TEU) of order (o,d,ω)	$(o, d, \omega) \in \mathcal{O}$
$T^T_{od\omega}$	- Transit time requirement (in weeks) for order (o,d,ω)	$(o, d, \omega) \in \mathcal{O}$

Modified variables

$f_{ijod\omega kr}$	- 1 if order (o,d,ω) is transported by a vessel of type k on arc (i,j) in route r, 0 otherwise
$t_{iod\omega r}$	- 1 if order (o,d,ω) is transhipped to route r in port i, 0 otherwise

Additional constraints

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(2 \cdot T_{ijk} \cdot f_{ijod\omega kr} + t_{iod\omega r}) \leq 2 \cdot T^T_{od\omega}, \qquad (o, d, \omega) \in \mathcal{O} \tag{A.72}$$

A.6 Flower transit time model

Below are all modifications and additions needed from the Basic transit time model to obtain the Flower transit time model. The additions and modifications needed from the Basic model to obtain the Flower model that are not affected by the transit time extension, are not included.

Modified variables

$t^I_{iod\omega r}$	- Internal transhipment of order (o, d, ω) in butterhub i of butterfly route r
$t^E_{iod\omega r}$	- External transhipment of order (o, d, ω) in port i of route r
$c_{ijod\omega kr}$	- 1 if the net flow in port i of order (o,d, ω) associated with ship type k and route r is non-positive, 0 otherwise.
$q_{imod\omega r}$	- The positive pairwise difference between out-flow and in-flow of order (o, d, ω) in neighbouring loops in butterfly route r

Modified constraints

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}}(2\cdot T_{ijk}\cdot f_{ijod\omega kr} + t^I_{iod\omega r} + t^E_{iod\omega r}) \leq 2\cdot T^T_{od\omega}, \qquad (o,d,\omega)\in\mathcal{O} \tag{A.73}$$

Additional constraints

$$q_{imod\omega r} \geq \sum_{k\in\mathcal{K}}(D_{od\omega}(f_{jmod\omega kr} - f_{ijod\omega kr}) - D_{od\omega}(4 - m_{jr} - p^N_{ijr} - x_{ijkr} - x_{jmkr}), \qquad (i,j),(j,m)\in\mathcal{A}, r\in\mathcal{R} \tag{A.74}$$

$$t^E_{iod\omega r} \geq \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} D_{od\omega}(f_{ijodkr} - f_{jiodkr}), \qquad i\in\mathcal{P}, i\neq o, i\neq d, (o,d,\omega)\in\mathcal{O}, r\in\mathcal{R} \tag{A.75}$$

$$t^I_{iod\omega r} \geq \sum_{j\in\mathcal{P}}\sum_{m\in\mathcal{P}} q_{jmod\omega r} - D_{od\omega}(1 - d_{ir}) - t^E_{iod\omega r}, \qquad i\in\mathcal{P}, (o,d,\omega)\in\mathcal{O}, r\in\mathcal{R} \tag{A.76}$$

A.7 Chain transit time model

Below are all modifications and additions needed from the Basic transit time model to obtain the Chain transit time model. The additions and modifications needed from the Basic model to obtain the Chain model that are not affected by the transit time extension, are not included.

Modified constraints

$$2 \cdot T_{ijk} \cdot \Big(\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} f_{ijod\omega kr} + \sum_{i\in\mathcal{N}\cap\bar{\mathcal{P}}}\sum_{j\in\mathcal{N}\cap\bar{\mathcal{P}}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}} f_{ijod\omega kr}\Big) + \sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}} t_{iod\omega r}\Big) \leq 2 \cdot T^{T}_{od\omega}, \quad (o,d,\omega)\in\mathcal{O} \tag{A.77}$$

Appendix B

Computational results

B.1 Testing of symmetry breaking constraints

B.1.1 Basic

Run time (in seconds) *(# ship types = 2 in all runs)*		# of demands		
# of ports	Model	**5**	**7**	**9**
5	B[ ]	9.04	8.82	43.21
	B[R1]	9.99	14.60	34.76
	B[R1+S]	13.34	21.20	32.13
	B[R2]	11.18	23.01	23.29
	B[R2+S]	14.16	17.29	33.97
	B[S]	10.12	23.35	27.27
7	B[ ]	15.70	50.32	157.23
	B[R1]	19.68	61.14	233.60
	B[R1+S]	38.37	87.25	230.96
	B[R2]	35.61	103.826	530.05
	B[R2+S]	52.74	94.766	193.52
	B[S]	16.64	53.86	467.80
9	B[ ]	2142.08	3242.70	6973.49
	B[R1]	2883.68	7339.31	*
	B[R1+S]	547.54	3502.59	*
	B[R2]	1165.54	7857.28	*
	B[R2+S]	738.64	3079.85	4317.74
	B[S]	3635.27	*	*

Table B.1: The table shows the results of the symmetry testing of the Basic model. The combination of SBCs with the lowest, second lowest and third lowest run time on a given instance is marked with shades of green. The squares marked with an * did not find the optimal solution within the cut off time of 10000 seconds.

B.1.2 Flower

Results symmetry testing Flower				
Instance	**Model with SBCs**	**# of nodes**	**Run time** (in seconds)	**Gap**
G[**525**]	F[]	5191	70.37	-
	F[L]	5319	84.24	-
G[**527**]	F[]	20075	134.62	-
	F[L]	35495	181.16	-
G[**529**]	F[]	20519	283.74	-
	F[L]	25341	265.01	-
G[**725**]	F[]	8347	219.44	-
	F[L]	1509	145.14	-
G[**727**]	F[]	377	151.03	-
	F[L]	3119	257.10	-
G[**729**]	F[]	14541	1113.73	-
	F[L]	7081	800.17	-
G[**925**]	F[]	65935	4127.09	-
	F[L]	60523	2690.03	-
G[**927**]	F[]	68454	10000*	11.42%
	F[L]	59891	10000*	2.40%
G[**929**]	F[]	37334	10000*	41.58%
	F[L]	31840	10000*	47.73%

Table B.2: Results of the symmetry testing of the Flower model. The least # of nodes, lowest run time and gap on a certain instance is colored in green. The cut off time was 10'000 on all runs. The run times on the instances that were not solved to optimality are marked with an *.

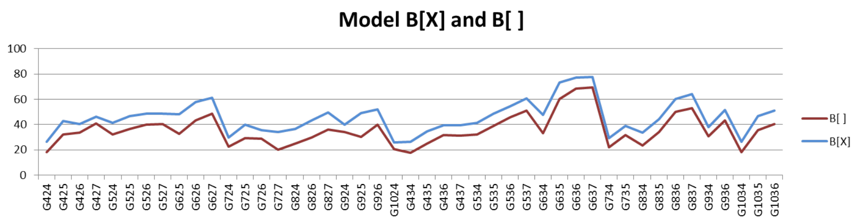
B.1.3 Chain

Results symmetry testing Chain			
Instance	**Model**	**# of nodes**	**Run time (in seconds)**
	C[ ]	26107	102.35
	C[T]	24107	99.28
G[524]	C[O]	27429	203.28
	C[U]	24569	144.71
	C[TO]	21829	109.79
	C[TOU]	13587	161.93
	C[ ]	8229	56.24
	C[T]	22467	138.83
G[525]	C[O]	24931	210.53
	C[U]	25063	178.06
	C[TO]	32055	484.59
	C[TOU]	35307	257.57
	C[ ]	18043	255.35
	C[T]	20257	321.78
G[526]	C[O]	22507	344.69
	C[U]	5011	128.22
	C[TO]	22270	414.51
	C[TOU]	32075	4953.04
	C[ ]	50143	373.81
	C[T]	59501	749.99
G[527]	C[O]	29641	450.81
	C[U]	41645	1193.01
	C[TO]	14319	366.20
	C[TOU]	22933	488.13
	C[ ]	62847	626.93
	C[T]	75145	646.86
G[529]	C[O]	74729	2071.80
	C[U]	68427	2365.48
	C[TO]	27237	295.77
	C[TOU]	43207	1433.04
	C[ ]	333971	2470.23
	C[T]	646151	9263.77
G[5211]	C[O]	163081	4361.85
	C[U]	325589	9726.02
	C[TO]	173641	5846.82
	C[TOU]	191895	8421.3
	C[ ]	443821	21987.60
	C[T]	208725	5557.69
G[5215]	C[O]	91887	5550.06
	C[U]	136935	7033.79
	C[TO]	55786	1792.29
	C[TOU]	92569	7627.48
	C[ ]	93858	959.90
	C[T]	91590	973.52
G[725]	C[O]	74366	976.46
	C[U]	46615	440.56
	C[TO]	37719	430.07
	C[TOU]	41181	1269.38
	C[ ]	9567	1144.88
	C[T]	15411	2656.19
G[728]	C[O]	10985	3101.29
	C[U]	10985	3101.29
	C[TOU]	7129	2436.99
	C[TOU]	3685	3456.85

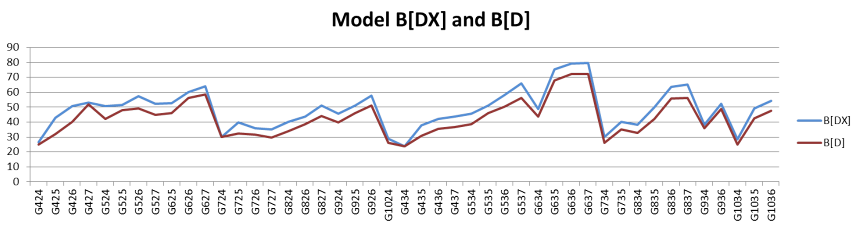
Table B.3: Results of the symmetry testing of the Chain model. The least # of nodes and lowest run time on a certain instance is colored in green. The cut off time was 30'000 on all runs, all instances were solved to optimality within the cut off time.

B.2 Testing of valid inequalities

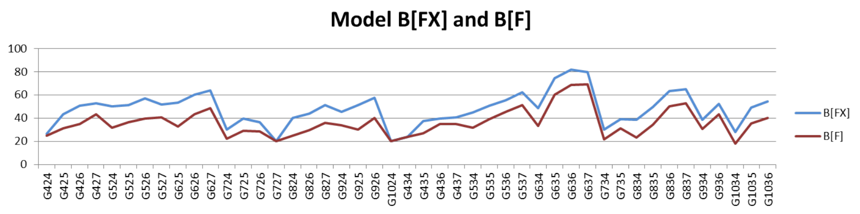
B.2.1 Xpress preprocessing



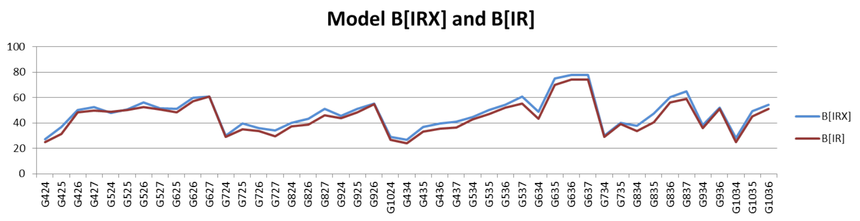
(a) LP objective values of B[X] and B[ ]



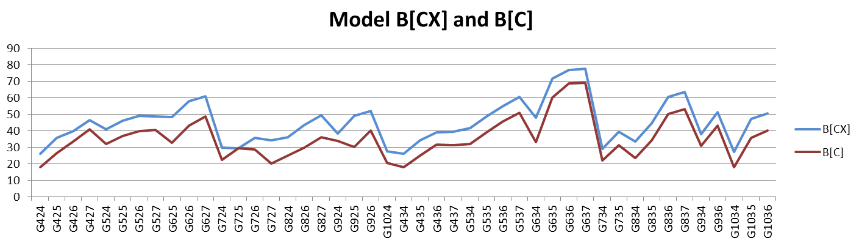
(b) LP objective values of B[DX] and B[D]



(c) LP objective values of B[FX] and B[F]

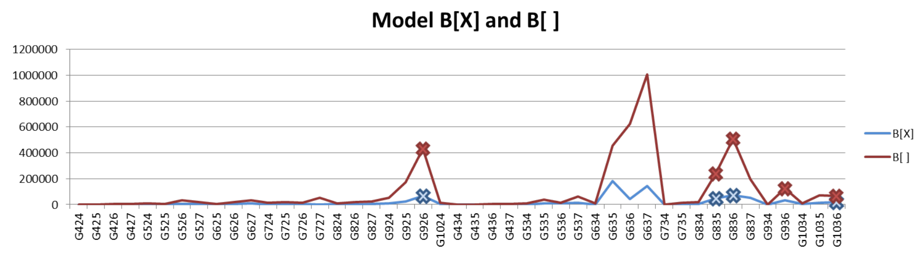


(d) LP objective values of B[IRX] and B[IR]

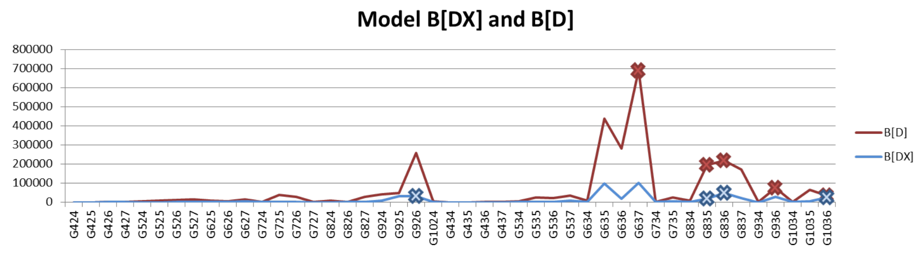


(e) LP objective values of B[CX] and B[C]

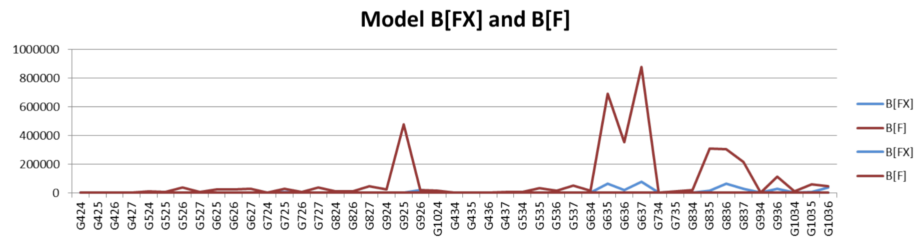
Figure B.1: **Xpress preprocessing's impact on LP objective values.** Figures B.1a to B.1e show how much the LP objective values on 45 instances improve in the Basic model by enabling Xpress preprocessing, in combination with the problem specific valid inequalities.



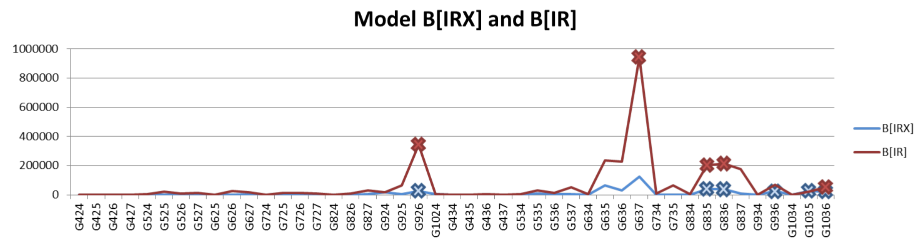
(a) # of nodes of B[X] and B[ ]



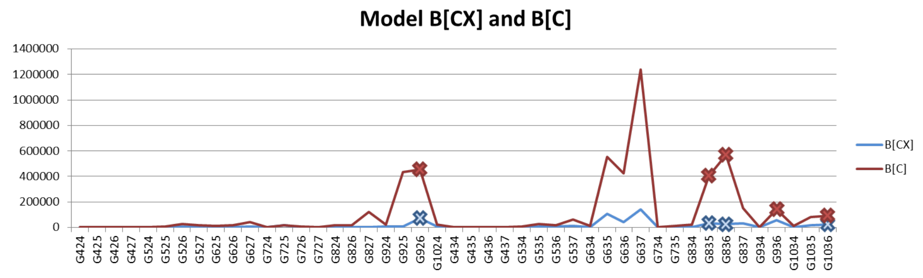
(b) # of nodes of B[DX] and B[D]



(c) # of nodes of B[FX] and B[F]

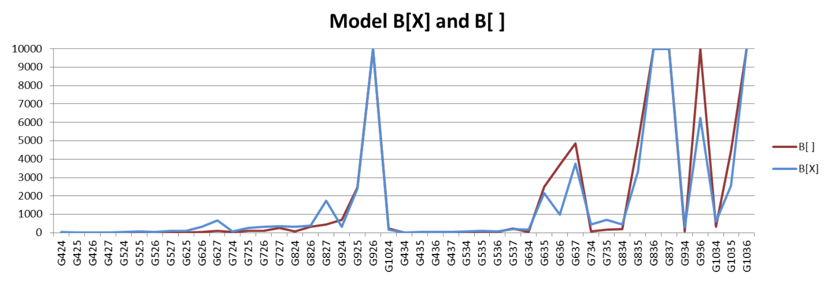


(d) # of nodes of B[IRX] and B[IR]

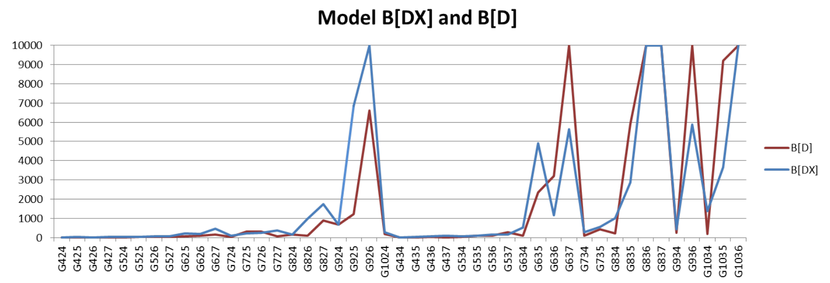


(e) # of nodes of B[CX] and B[C]

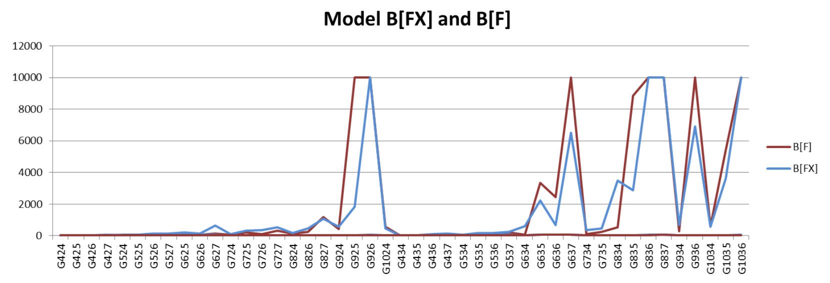
Figure B.2: **Xpress preprocessing's impact on number of nodes.** Figures B.8a to B.2e show how the number of nodes decrease in the Basic model by enabling Xpress preprocessing, in combination with the problem specific valid inequalities.



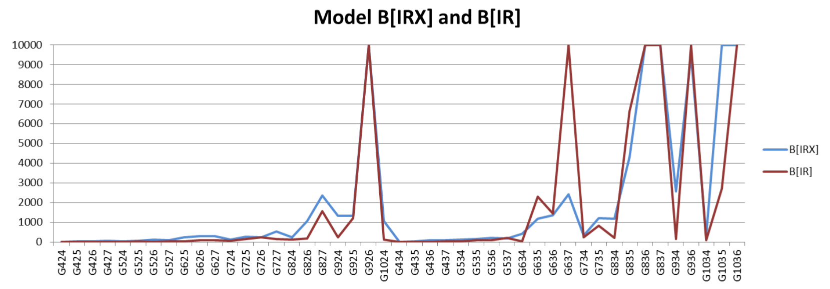
(a) Run times of B[X] and B[ ]



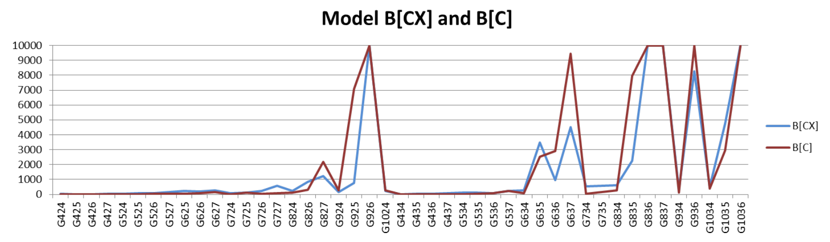
(b) Run times of B[DX] and B[D]



(c) Run times of B[FX] and B[F]



(d) Run times of B[IRX] and B[IR]



(e) Run times of B[CX] and B[C]

Figure B.3: **Xpress preprocessing's impact on run times.** Figures B.3a to B.3e show how the run times decrease in the Basic model by enabling Xpress preprocessing, in combination with the problem specific valid inequalities.

B.2.2 Integer rounding: Choosing the best α-values

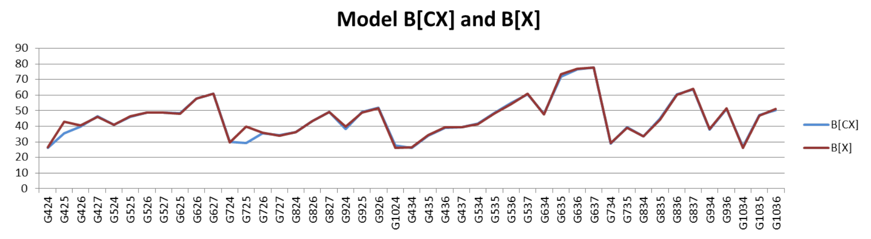
	B[CX]				**B[C]**			
	G[434]	**G[435]**	**G735**	**Average**	**G[434]**	**G[435]**	**G[735]**	**Average**
Alt 1								
LP	1	2	1	*1.3*	1	1	1	*1*
#n	2	2	2	*2*	2	2	1	*1.7*
R.t.	3	2	2	*2.3*	3	2	2	*2.3*
Average	*2*	*2*	*1.7*	*1.9*	*2*	*1.7*	*1.3*	*1.7*
Alt 2								
LP	2	1	2	*1.7*	1	1	1	*1*
#n	1	1	3	*1.7*	1	1	2	*1.3*
R.t.	1	1	1	*1*	1	1	3	*1.7*
Average	*1.3*	*1*	*2*	*1.4*	*1*	*1*	*2*	*1.3*
Alt 3								
LP	1	2	1	*1.3*	1	1	1	*1*
#n	2	2	1	*1.7*	2	2	1	*1.7*
R.t.	2	3	3	*2.7*	2	3	1	*2*
Average	*1.7*	*2.3*	*1.7*	*1.9*	*1.7*	*2*	*1*	*1.6*

Table B.4: Analysis of LP solution (LP), the number of nodes (#n) and the run time (R.t.) with different α values for the CM + IR-valid inequalities. In alternative 1 α_1 is set to 1.001, while alternative 2 has α_1 and α_2 set to 2.001, while alternative 3 has α_1, α_2 and α_3 set to 3.001. The rankings are made comparing each instance for a cut/α-combination, where the best one(s) has/have been ranked as 1, the second as 2 etc. There are calculated different averages for each alternative and cut combination: Over the different parameters for each instance, over different instances for each parameters and also the average of these two averages. This information have been used to analyse emphasizing either an instance size, a parameter or the α-alternative and cut-combination as a whole.

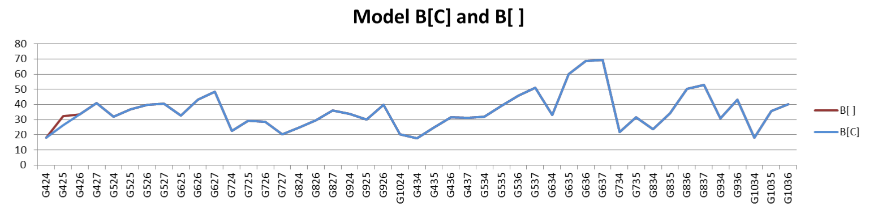
	B[IRX]				**B[IR]**			
	G[434]	**G[435]**	**G[735]**	**Average**	**4**	**5**	**7**	**Average**
Alt 1								
LP	2	2	2	*2*	1	2	2	*1.7*
#n	2	2	2	*2*	1	2	1	*1.3*
R.t.	3	3	1	*2.3*	1	3	1	*1.7*
Average	*2.3*	*2.3*	*1.7*	*2.1*	*1*	*2.3*	*1.3*	*1.6*
Alt 2								
LP	1	1	1	*1*	1	1	1	*1*
#n	1	1	1	*1*	2	1	2	*1.7*
R.t.	1	2	2	*1.7*	3	2	3	*2.7*
Average	*1*	*1.3*	*1.3*	*1.2*	*2*	*1.3*	*2*	*1.8*
Alt 3								
LP	1	1	1	*1*	1	1	1	*1*
#n	1	1	1	*1*	2	1	2	*1.7*
R.t.	2	1	3	*2*	2	1	2	*1.7*
Average	*1.3*	*1*	*1.7*	*1.3*	*1.7*	*1*	*1.7*	*1.4*

Table B.5: The table shows the results from runs of the D/C + IR-valid inequalities with different alpha-values. In alternative 1 α_1 is set to 1.001, while alternative 2 has α_1 and α_2 set to 2.001, while alternative 3 has α_1, α_2 and α_3 set to 3.001. The rankings are made comparing each instance for a cut/α-combination, where the best one(s) has/have been ranked as 1, the second as 2 etc. There are calculated different averages for each alternative and cut combination: Over the different parameters for each instance, over different instances for each parameters and also the average of these two averages. This information have been used to analyse emphasizing either an instance size, a parameter or the α-alternative and cut-combination as a whole.

B.2.3 Capacity miles

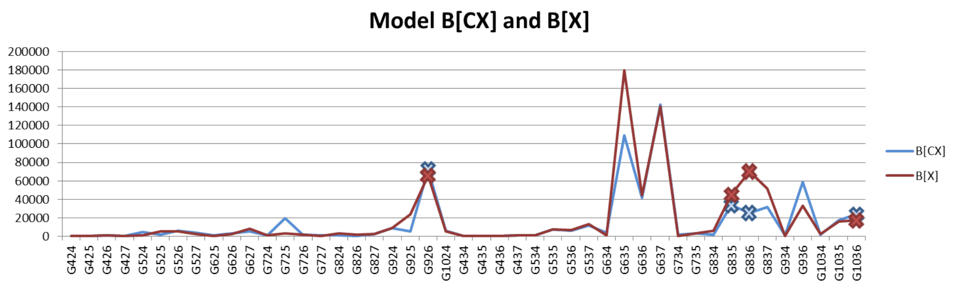


(a) LP objective values of B[CX] and B[X]

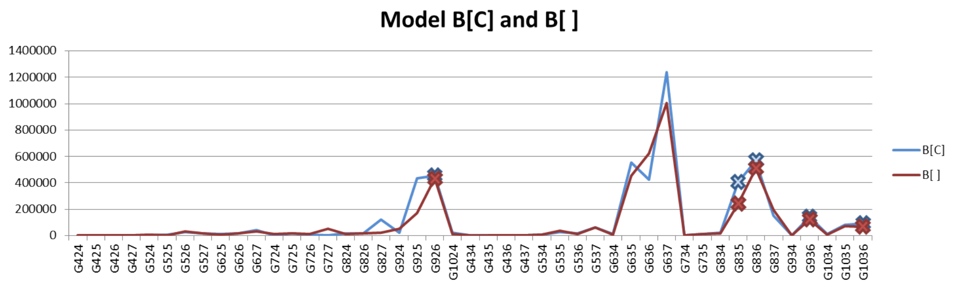


(b) LP objective values of B[C] and B[ ]

Figure B.4: **Capacity miles' impact on LP objective values.** Figures B.4a and **??** show how the LP objective values on 45 instances vary when the CM valid inequalities are included and not in the Basic model.

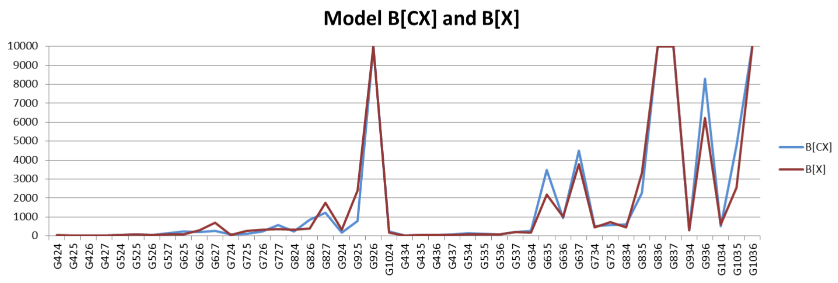


(a) # of nodes of B[IRX] and B[X]

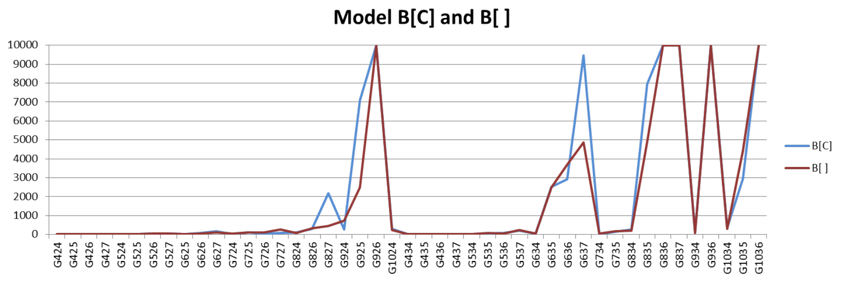


(b) # of nodes of B[IR] and B[ ]

Figure B.5: **Capacity miles' impact on number of nodes.** Figures B.5a and B.5b show how the number of nodes on 45 instances vary when the CM- valid inequalities are included and not in the Basic model.



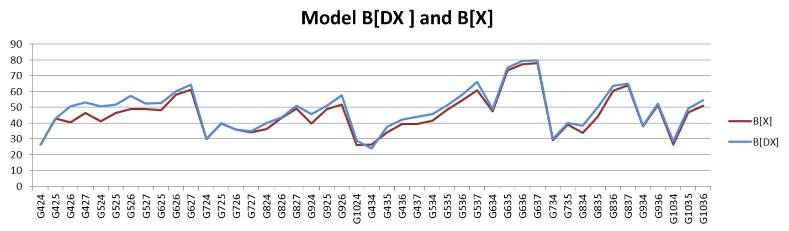
(a) Run times of B[FX] and B[X]



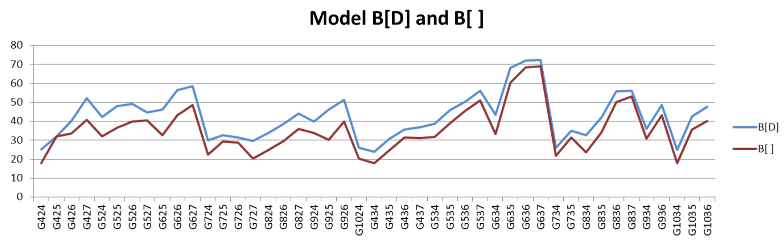
(b) Run times of B[F] and B[ ]

Figure B.6: **Capacity miles' impact on run times.** Figures B.6a and B.6b show how the number of nodes on 45 instances vary when the CM- valid inequalities are included and not in the Basic model.

B.2.4 D/C valid inequalities

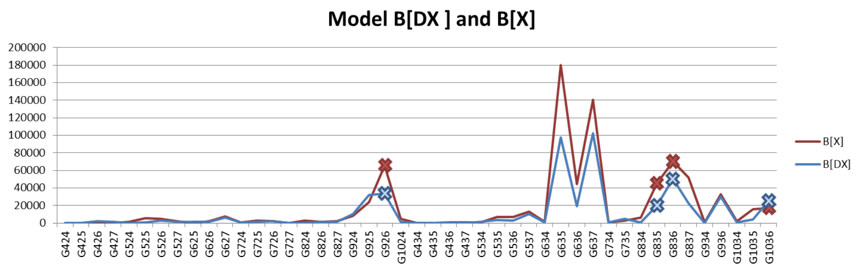


(a) LP objective values of B[DX] and B[X]

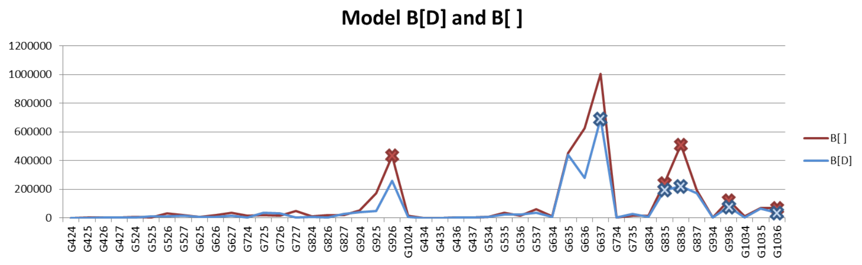


(b) LP objective values of B[D] and B[ ]

Figure B.7: **D/C valid inequalities' impact on LP objective values.** Figures B.7a and B.7b show how the LP objective values on 45 instances vary when the D/C valid inequalities are included and not in the Basic model. The blue line is when Xpress preprocessing is enabled, and the red line is when it is turned off.

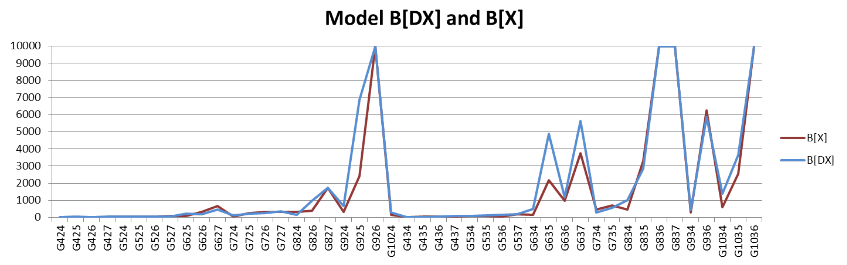


(a) # of nodes of B[X] and B[ ]

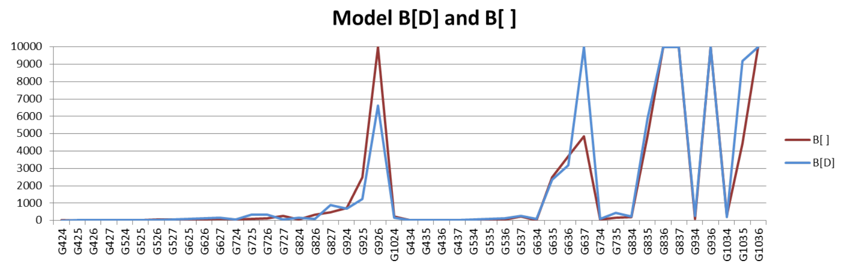


(b) # of nodes of B[DX] and B[D]

Figure B.8: **D/C valid inequalities' impact on number of nodes.** Figures B.7a and B.7b show how the number of nodes on 45 instances vary when the D/C valid inequalities are included and not in the Basic model. The blue line is when Xpress preprocessing is enabled, and the red line is when it is turned off.



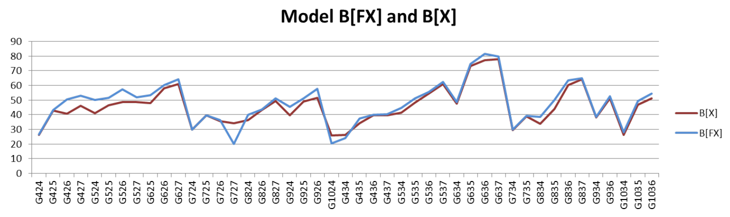
(a) Run times of B[X] and B[ ]



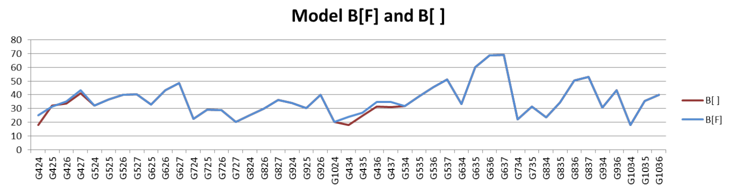
(b) Run times of B[DX] and B[D]

Figure B.9: **D/C valid inequalities' impact on run times.** Figures B.9a and B.9b show how the number of nodes on 45 instances vary when the D/C valid inequalities are included and not in the Basic model. The blue line is when Xpress preprocessing is enabled, and the red line is when it is turned off.

B.2.5 Flow valid inequalities

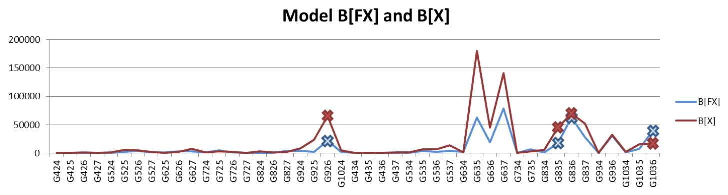


(a) LP objective values of B[FX] and B[X]

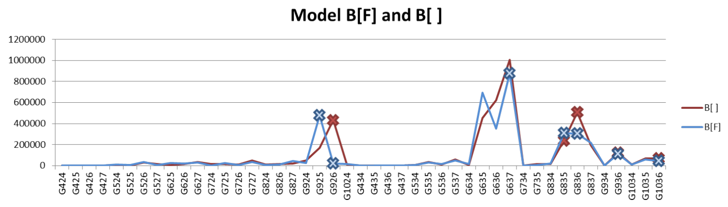


(b) LP objective values of B[F] and B[ ]

Figure B.10: **D/C flow-valid inequalities' impact on LP objective values.** Figures B.10a and B.10b show how the LP objective values on 45 instances vary when the D/C flow-valid inequalities are included and not in the Basic model.

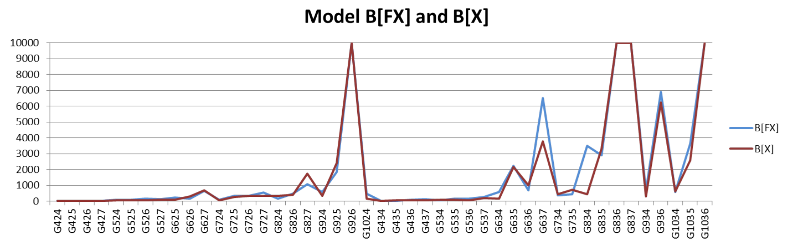


(a) # of nodes of B[FX] and B[X]

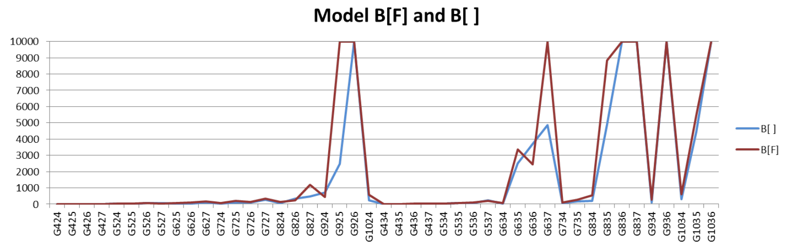


(b) # of nodes of B[F] and B[ ]

Figure B.11: **D/C flow-valid inequalities' impact on number of nodes.** Figures B.11a and B.11b show how the number of nodes on 45 instances vary when the D/C flow- valid inequalities are included and not in the Basic model.



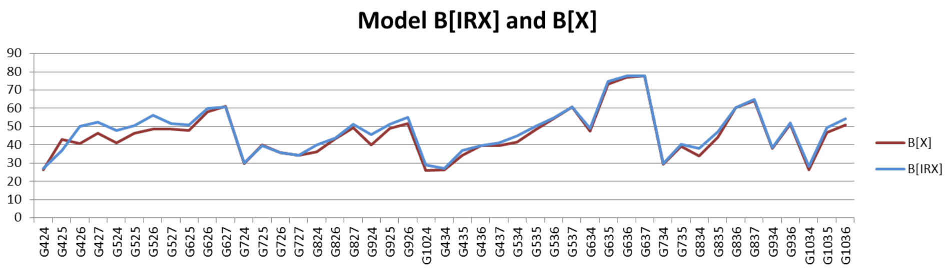
(a) Run times of B[FX] and B[X]



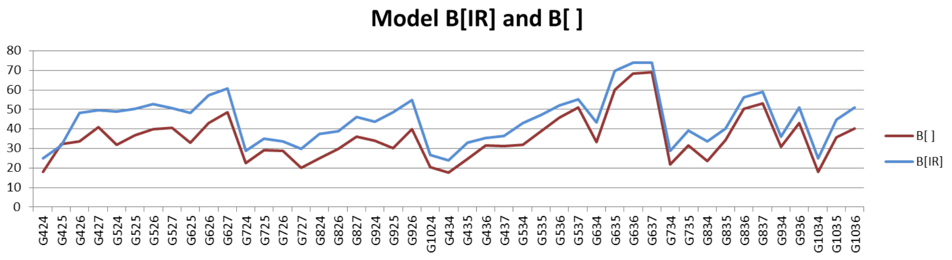
(b) Run times of B[F] and B[ ]

Figure B.12: **D/C valid inequalities' impact on run times.** Figures B.12a and B.12b show how the number of nodes on 45 instances vary when the D/C flow- valid inequalities are included and not in the Basic model.

B.2.6 Demand/capacity with integer rounding

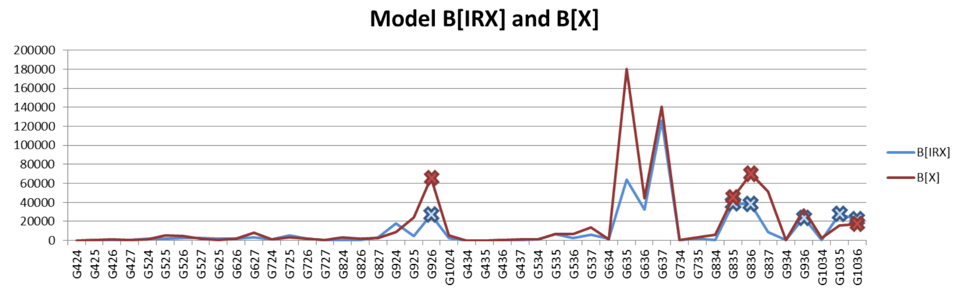


(a) LP objective values of B[IRX] and B[X]

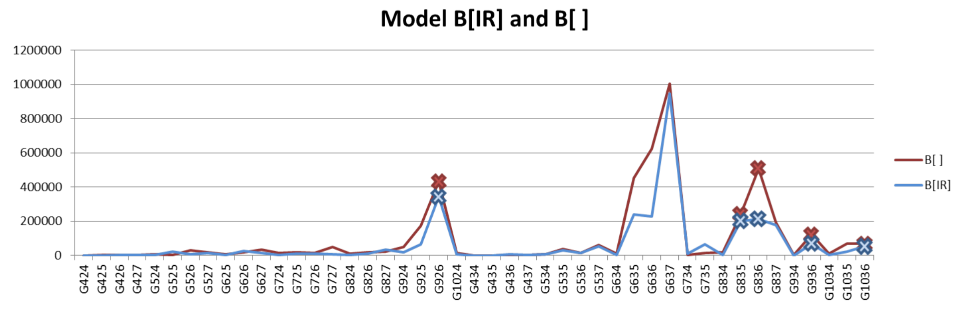


(b) LP objective values of B[IR] and B[ ]

Figure B.13: **D/C IR-valid inequalities' impact on LP objective values.** Figures B.13a and B.13b show how the LP objective values on 45 instances vary when the D/C IR- valid inequalities are included and not in the Basic model.

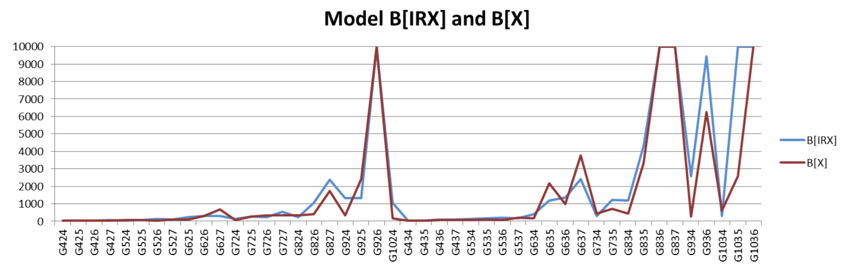


(a) # of nodes of B[IRX] and B[X]

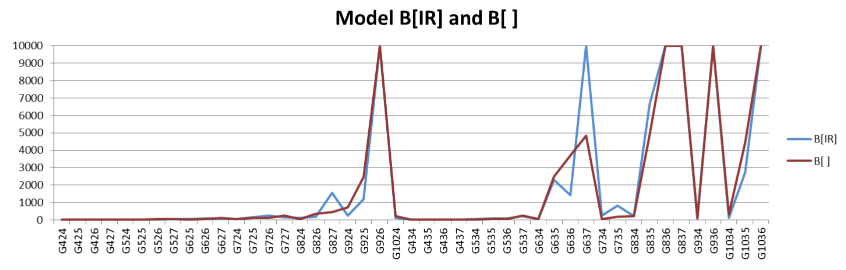


(b) # of nodes of B[IR] and B[ ]

Figure B.14: **D/C IR-valid inequalities' impact on number of nodes.** Figures B.11a and B.11b show how the number of nodes on 45 instances vary when the D/C IR- valid inequalities are included and not in the Basic model.



(a) Run times of B[FX] and B[X]



(b) Run times of B[F] and B[ ]

Figure B.15: **D/C IR valid inequalities' impact on run times.** Figures B.15a and B.15b show how the number of nodes on 45 instances vary when the D/C IR- valid inequalities are included and not in the Basic model.

Comparison of the LP objective values of the model variant

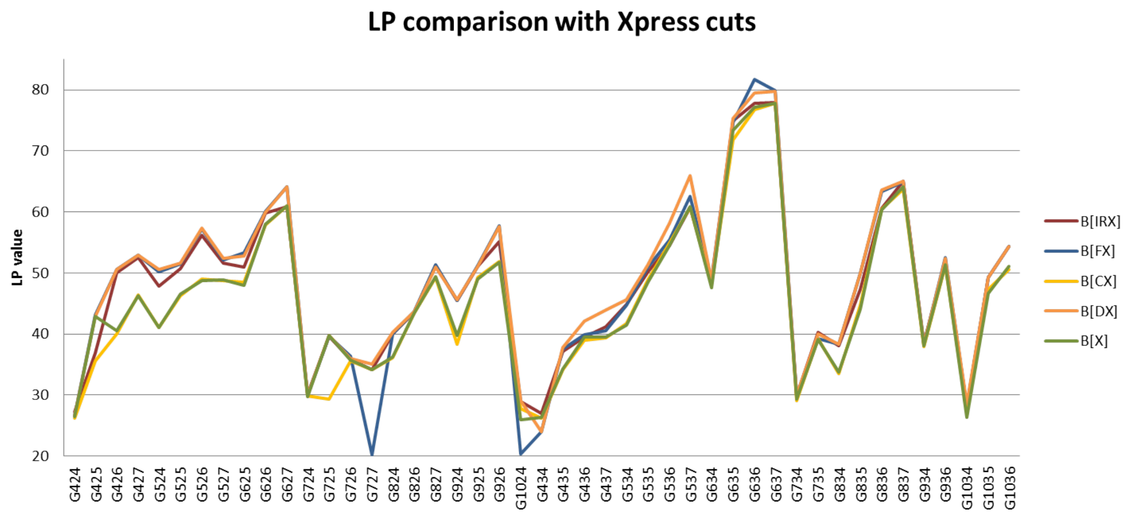


Figure B.16: The LP objective values of the models with Xpress-valid inequalities (B[X], B[DX], B[FX], B[IRX] and B[CX]).

Instance categorization

	# of ports	**Small**	**Medium**	**Large**
	4	G[424]-G[427]		
	5	G[524]-G[527]		
# of shiptypes= 2	6	-	G[625]-G[627]	
	7		G[724]-G[727]	
	8		G[824]	G[825]-G[827]
	9			G[924]-G[926]
	10			G[1024]
	4	G[434]-G[435]	G[436]-G[437]	
	5	G[534]	G[535]-G[537]	
# of shiptypes= 3	6			G[634]-G[637]
	7			G[734]-G[737]
	8			

Table B.6: Instance categorization

Appendix C

Attachments

Included as a ZIP file can be found:

1. Mosel code for the Basic model, Flower model, Chain model, Basic model with transit time, Flower model with transit time and Chain model with transit time.
2. The data instances used for the technical computational study