Routing and scheduling in liner shipping with multi-start local search heuristics

S.A. Lachner
276391

V.A.G. Boskamp
305434

Supervised by Prof.dr.ir. Rommert Dekker
Co-reader: Prof.dr.ir. Uzay Kaymak

Master Thesis of Economics & Informatics, Computational Economics,
Econometric Institute, Erasmus School of Economics,
Erasmus University Rotterdam, the Netherlands

January 19, 2011

Abstract

The continuous growth in international container traffic volumes makes it ever more important for carriers to optimize their service network. In this thesis, we present a multi-start local search algorithm for solving the routing and scheduling problem in liner shipping. The objective is to find a service network of routes, given the demand between ports, that maximizes profit.

The algorithm consists of a randomized initialization phase that generates initial networks, and a local search phase that tries to improve the solution using local search operators. For each phase we present different implementations, such that several algorithm configurations are obtained, representing different multi-start local search heuristics. For the first phase, we propose the *quantity sort* insertion heuristic, and the *profit-driven sort* insertion heuristic. For the second phase, we propose three local search operators. The *route-length operator* removes ports from round trips that incur more costs than revenue, and tries to allocate unassigned cargoes by adding ports to round trips. The *port-exchange operator* relocates ports within a route or between routes in an attempt to improve solutions. The *transhipment operator* introduces the use of hubs and transhipment to save costs and allocate the remaining cargoes.

The different configurations are subjected to an extensive benchmark study in order to analyze their performance. To that end, we also propose a data set that is based on the actual Asia-Europe network of Maersk Line. Results indicate that running the route-length operator, followed by the port-exchange operator, on average yields the best networks. The sorting method turns out to have negligible influence on the final result. We achieve a gap to the upper bound of less than 5.5%. When we use a heterogeneous fleet, the gap is less than 2.6%.

Table of contents

List of Figures **7**

List of Tables **9**

1 Introduction **11**

1.1 Background . . . 11

1.2 Research goal . . . 12

1.3 Research questions and objectives . . . 14

1.3.1 Hypotheses . . . 15

1.3.2 Objectives . . . 15

1.4 Methodology . . . 15

1.5 Structure . . . 16

2 Liner shipping **19**

2.1 Introduction to liner shipping . . . 19

2.2 Routing and scheduling problem . . . 24

2.2.1 Mathematical programming approaches . . . 24

2.2.2 Heuristic approaches . . . 25

2.2.3 Routing and scheduling in reality . . . 27

2.3 Related problems . . . 28

2.3.1 Other routing problems . . . 28

2.3.2 Fleet sizing problem . . . 30

2.3.3 Master bay planning problem . . . 30

2.3.4 Empty container repositioning . . . 31

2.3.5 Transhipment hub locations . . . 31

2.4 Conclusion . . . 32

3 Formulation of the routing and scheduling problem in liner shipping 35
3.1 Assumptions 35
3.2 Problem description 38
3.2.1 Costs 38
3.2.2 Revenue 40
3.2.3 Consequences of chosen cost-revenue structure 40
3.2.4 Concluding remarks 42

4 Multi-start local search heuristics 43
4.1 Introduction 43
4.2 Phase 1: Generating initial solutions 44
4.2.1 Quantity Sort 48
4.2.2 PDA Sort 49
4.3 Phase 2: Local search optimization 52
4.3.1 Sample service network 53
4.3.2 Route-length operator 53
4.3.3 Port-exchange operator 56
4.3.4 Transhipment operator 62
4.4 Conclusion 66

5 The Maersk Asia-Europe data set 69
5.1 Introduction 69
5.2 Data set description 72
5.2.1 Original service network 72
5.2.2 Ports 73
5.2.3 Fleet 73
5.2.4 Demand 74
5.2.5 Distances 77
5.2.6 Revenues & Costs 77
5.3 Conclusion 77

6 Experimental design & results 79
6.1 Introduction 79
6.2 Experimental Design 80
6.2.1 Model variables 81
6.2.2 Algorithmic parameters 85
6.2.3 Data set 87

6.2.4 Implementation 88
6.2.5 Hypotheses 89
6.3 Benchmark Results 91
6.3.1 Interpretation 96
6.3.2 Best network 98
6.3.3 Optimality & Upper bound 104
6.4 Result Analysis 106
6.4.1 Analysis design 107
6.4.2 Analysis 110
6.5 Conclusion 117

7 Additional experiments 123
7.1 A seven-port model of the Asia-Europe trade lane 124
7.1.1 Assumptions and model variables 124
7.1.2 Results 125
7.2 Different transhipment operator implementations 130
7.2.1 Transhipment operator implementations 130
7.2.2 Experimental design 133
7.2.3 Results 134
7.3 Using smaller sized ships 137
7.3.1 Using a fixed ship capacity of 8,000 TEU 137
7.3.2 Optimizing individual routes using a mixed fleet 139
7.4 Varying multi-start initializations 144
7.5 Re-applying local search operators 146
7.6 Conclusion 148

8 Conclusions and Future Research 151
8.1 Conclusions 152
8.2 Future Research 155

9 Discussion 157

Glossary 161

Bibliography 163

A Algorithms 171

B Maersk Asia-Europe data set **185**

C Best network **195**

List of Figures

2.1 Mitsui O.S.K. Lines (MOL) - AEX Service. 20

4.1 Outline of multi-start local search algorithm 45
4.2 Map of sample ports 54
4.3 Example operations of the route-length operator 56
4.4 Example operations of the port-exchange operator 61
4.5 Direct path versus transhipment path 66

5.1 Difference in 6,500 TEU and 14,770 TEU vessel size 73

6.1 Historical freight rates on the Asia-Europe trade lane 84
6.2 Profit distribution across all 2,100 networks 94
6.3 Sample route of the best network 103

7.1 Profits of P/Q & 2 and Q & 1 configurations compared 129
7.2 Network profits before and after local search 146

B.1 Map of ports in MAE data set 193

List of Tables

2.1 Mitsui O.S.K. Lines (MOL) - AEX service schedule 20

4.1 Services of the sample service network 54

6.1 Model variables and algorithmic parameters 87
6.2 Results from the benchmark simulation 92
6.3 Average profits with different sorting methods 93
6.4 Average improvement compared to phase I 93
6.5 Various operator sequences for the PQ & 1-2-3 configuration 94
6.6 Average execution times of multi-start local search phases per network . . 95
6.7 Detailed results of the best network 99
6.8 Performance of best network versus upper bound 106
6.9 Using the F-test 108
6.10 Different configurations: F-ratios 111
6.11 Different configurations: p-values 112
6.12 Two sorting methods: F-ratios 113
6.13 Different configurations: p-values 113
6.14 Different operator sequences for P/Q: F-ratios 114
6.15 Different operator sequences for P/Q: p-values 115
6.16 MLR: overview of features and their regression coefficients 116

7.1 Demand matrix of the seven-port model 125
7.2 Adjusted model variable settings 126
7.3 Algorithms' results of the 7-port model 127
7.4 Multi-start local search algorithm: results of all configurations 128
7.5 Results of the transhipment operator implementations 134
7.6 Results of the Q & 1-2 configuration using 8,000 TEU ships 138
7.7 Using mixed ship sizes: costs per capacity 141

7.8 Result of using mixed ship sizes 143
7.9 Best results of the different multi-start initializations 145
7.10 Results of re-applying local search operators 147

A.1 Overview of algorithm pseudocodes 171

B.1 Services in the original Maersk Asia-Europe network 186
B.2 Vessels operating in the original Maersk Asia-Europe network 187
B.3 Ports in the MAE data set 188
B.4 Vessels in the MAE data set 189
B.5 Demand matrix of the MAE data set 190
B.6 Distance matrix of the MAE data set 191
B.7 Natural order of ports in the MAE data set 192

C.1 Services in our best service network 196
C.2 Comparison of port occurrence in Maersk network & our best network . . 197

Chapter 1

Introduction

1.1 Background

Despite the global economic downturn of the last years, international seaborne trade continues to grow. According to the United Nations Conference on Trade and Development (UNCTAD 2009), the volume of international seaborne trade increased with 3.6 percent to 8.17 billion tons in 2008. The world merchant fleet expanded by 6.7 percent to 1.19 billion dead-weight tons.

In maritime transportation we differentiate between three types of shipping, industrial shipping, tramp shipping, and liner shipping (Lawrence 1972). Industrial shipping is characterized by the cargo owner controlling the ship, and shipping takes place only for the company itself. Tramp ships have no fixed schedule and trade on the spot market, following the available cargo and making deals for shipments. Liner shipping companies provide services according to a regular repetitive schedule, similar to a bus line. These three types are not mutually exclusive, since shipping companies may operate ships for different types of operations (Christiansen et al. 2004). In this work we focus on liner shipping and consider standardized containers as cargo.

Container ships operate on scheduled *routes*, visiting several ports in a closed *cycle*, i.e., returning to the starting port. The sequence of ports visited in the cycle is referred to as the *string of ports*. Other words for cycle are *loop*, *pendulum*, and *service*. Routes in liner shipping are *main-line* routes, connecting the largest ports (i.e., *hubs*) using big container ships. In contrast, feeder lines provide regional services on *feeder* routes, feeding and distributing containers to and from the hub using smaller-sized ships like barges. A liner shipping carrier usually has a global service network, consisting of several main-line loops between multiple continents. The schedule of each service is fixed for longer time periods, and consists of regular port calls. That is, container ships depart at least once a week from each port, on a given day of the week. The number of departures from a port for a specific cycle is called the service *frequency*, which is seen as an important criterion for customer satisfaction and provides for competition among carriers.

Liner shipping companies face decision problems at three different planning levels, the long term strategic planning level, the medium term tactical planning level, and the short term operational planning level (Agarwal & Ergun 2008). Strategic planning deals with determining the optimal fleet size and composition. Tactical planning covers the design of the service network, i.e. determining the set of routes that maximizes revenue. Operational planning considers choosing which cargo to transport, and assigning the cargo to the routes. These planning levels are highly related, making the decision problems very complex. The routes designed in the tactical planning level should be based on the demand in ports, and the fleet composition should be able to provide customers with a regular repetitive schedule on these routes. On the other hand, the fleet composition puts restrictions on which routes to service and the amount of cargo to be handled.

1.2 Research goal

The problem considered in this work is the routing and scheduling problem in liner shipping. Routing refers to the sequence of ports that ships visit, whereas scheduling is routing with time windows in which the delivery must take place (Ronen 1983). Regarding the planning levels, we focus on the tactical planning and the operational planning level, also known as the simultaneous ship-scheduling and cargo-routing problem when following (Agarwal & Ergun 2008). Given the yearly demand between ports, our objective is to find a weekly schedule and a set of routes for a fleet of vessels that maximizes profit. Routing and scheduling are most often considered together because the regular visit of a port already implies a schedule.

Designing a service network consists of creating routes and allocating demand over these routes. The number of candidate service networks grows exponentially with the number of ports considered. Ideally, calculating the objective function for all feasible service networks would reveal the optimal solution. However, the problem is NP-hard, making it extremely time-consuming to find the optimal solution.

To address this problem, one can impose constraints that limit the solution space. For example, one could generate all possible routes, and select only the most profitable ones to compose a service network. Another solution can be found in randomized initialization. This can reduce computation time dramatically, while the solution can approach optimality if the method is used properly. Randomized initialization is often repeated many times to create a diverse starting set, at which point the method is usually called *multi-start*. Local search methods can then be used to search neighbouring solutions, and find the best solution for each initialization. The multi-start characteristic must prevent ending up in a local optimum. Until now, this type of algorithm has only been applied in industrial and tramp shipping.

The relevance of good solutions to the routing and scheduling problem in liner shipping is evident. Carriers are always optimizing their service network and schedules to maximize profit. Service network design has transformed to a customer-oriented differentiation exercise among carriers, and the companies are in the process of reviewing their network strategy (Notteboom 2004). Even today, many carriers still have planners that perform routing and scheduling manually based on their professional knowledge and experience (Lam 2009). However, tools exist to support the routing and scheduling in sea shipping. One of the heuristic-based systems actually used by planners in the industry is TurboRouter, albeit for industrial and tramp shipping operations. TurboRouter uses the multi-start local search heuristic described in (Fagerholt & Lindstand 2007).

In this work we will transform and apply the multi-start local search heuristic that is used in TurboRouter to liner shipping operations. The nature of the algorithm allows for variation among its two main phases, where phase 1 is an *insertion heuristic* and phase 2 is a *local search heuristic*. We will design different implementations for each phase. The implementations of the insertion heuristic are referred to as *sorts*, and the implementations of the local search heuristic are referred to as *operators*. For the first phase we will implement the *quantity sort* insertion heuristic from (Brønmo et al. 2007), and a *profit-driven sort* insertion heuristic. For the second phase we propose three new local search operators, the *route-length* operator, the *port-exchange* operator, and the *transhipment* operator. By combining the implementations of each phase we obtain different algorithms that will be used in a benchmark simulation. The goal is to find the most effective algorithm, i.e., the combination of implementations that performs best.

The quality of a benchmark depends for a large part on the data set and input parameters that are used. Since there is no de facto standard data set available in the field, a considerable part of this work will focus on creating such a data set. Besides creating a data set and performing a benchmark study, we provide an extensive analysis of the results, and discuss the resulting service networks based on their characteristics. In addition, we perform additional experiments that justify the experimental design of the benchmark study.

1.3 Research questions and objectives

In this research we will implement multi-start local search algorithms for solving the routing and scheduling problem in liner shipping. The motivation for choosing the multi-start local search approach lies in its previous successful application in the industrial and tramp shipping industry. The main research question can be formulated as follows.

- *What is the performance of multi-start local search heuristics for solving routing and scheduling problems in liner shipping?*

In order to answer the main research question, we decompose it into subquestions. Each subquestion contributes to the knowledge neccessary to answer the main research question. The subquestions that are related directly to the main research question are the following.

1. *What are the specifics of liner shipping and what are its typical routing and scheduling problems?*
2. *What are multi-start and local search techniques, and why do they work?*
3. *Can we transform the multi-start local search algorithm from tramp shipping studies to the case of liner shipping?*

The main research question cannot be tackled without answering the above subquestions. In addition, the following subquestions emerge from preliminary research on routing and scheduling and multi-start local search heuristics.

4. *What is an adequate data set to assess the effectiveness of solutions to the problem?*
5. *Does the more advanced profit-driven sort insertion heuristic contribute more to the quality of the solutions in the multi-start heuristic than the simple quantity sort insertion heuristic?*
6. *What local search operators contribute the most to the objective function?*
7. *What is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping?*
8. *Under what circumstances is the use of transhipment hubs effective?*

Together, these subquestions represent the general outline of the thesis.

1.3.1 Hypotheses

To our knowledge, the multi-start local search approach has not been applied to liner shipping operations before. We believe that with the right adjustments, it can prove to be successful in liner shipping operations as well. Overall, we expect the profit-driven sort insertion heuristic to yield the highest quality of initial solutions, since an early focus on profits should help in the process of maximizing revenue. The local search operator we expect to contribute to the objective function the most is the transhipment operator. The rationale behind this expectation is the use of transhipment hubs in reality. Transhipment makes the model much more complex, but should increase the total cost-effectiveness of a service network.

1.3.2 Objectives

In short, the objectives and contributions of this research are: (1) to propose multi-start local search heuristics for the routing and scheduling problem in liner shipping, (2) to add an analysis of the effectiveness of different implementations of these heuristics, (3) to propose a data set that combines general applicability and reality, and (4) to investigate under what circumstances the use of transhipment hubs is effective. This research will also uncover possible future work directions in this field.

1.4 Methodology

In order to answer the research questions, we use the following methodology. First, we introduce the reader to the world of liner shipping, and present an extensive literature review on the routing and scheduling problem, as well as related problems. More specifically, we perform an *analysis* of successful planning concepts from related work. Then, by *programming*, we form a *synthesis* of different multi-start local search concepts. These heuristics serve as an input to a *computer experimentation*, that benchmarks their performance with respect to effectiveness and efficiency. When referring to effectiveness we mean the profit that a service network yields, and when referring to efficiency we target the execution times of the algorithms. Naturally, we want to maximize profit, and minimize the execution times.

In order to perform the experiment, we need a data set that will provide us with realistic cases. To come up with a decent data set we collect data from multiple secondary sources, mainly originating from previous work on the problem. We will analyse these sources, and discuss the contents of a good data set for the routing and scheduling problem in liner shipping. Using the available data, we *construct* a data set that serves a general purpose, i.e. the data set should fit our problem, but will be usable to other researchers as well.

During the experiment, a large number of simulations will be run in order to draw reliable conclusions. To obtain unbiased results, several configurations are simulated, where every configuration consists of a different combination of insertion and local search heuristics. After the experiment, we perform an *analysis* of the results, the variables, and their influence on the performance of the algorithms. In addition, the best network from the experiment is presented and subjected to further investigation.

To support the results following from the main experiment, we perform various additional experiments. In order to place the performance of our approach into perspective, we will benchmark the multi-start local search algorithm against algorithms from previous work. Furthermore, we investigate different approaches to model transhipment hubs. We also provide additional experiments with varying model variables and parameter settings, in order to get more insight into their influence on the final performance. The implementation of the algorithms, as well as the benchmark simulation and the additional experiments will be performed in Matlab.

1.5 Structure

The remainder of this thesis is organized as follows.

In Chapter 2, we present an overview of the literature that relates to our research problem. The goal is to answer the first two research subquestions, *what are the specifics of liner shipping and what are its typical routing and scheduling problems?*, and *what are multi-start and local search techniques, and why do they work?* First, we give a general introduction to liner shipping. Then, we discuss the routing and scheduling problem and existing solution approaches. Furthermore, we present other problems related to routing and scheduling as well as liner shipping in general.

In Chapter 3, we give the problem formulation that is used in this research. The routing and scheduling problem occurs in many different forms, since variables can be considered as either in or out of scope. Therefore, we state the assumptions of our problem variant, and give the precise problem description. Furthermore, we discuss important problem variables as part of the problem description.

Chapter 4 presents the heuristic multi-start local search algorithms. The objective is to answer research subquestion 3, *can we transform the multi-start local search algorithm from tramp shipping studies to the case of liner shipping?* The multi-start local search algorithms consist of two phases. In the first phase, initial solutions are generated, and in the second phase these solutions are improved using local search. We give a detailed description of the different implementations for both phases.

In Chapter 5, we investigate data sets from related work and discuss their usability for our problem. The goal is to answer research subquestion 4, i.e., to find out *what is an adequate data set to assess the effectiveness of solutions to the problem?* Apart from reviewing existing data sets, we present a new data set that perfectly fits our objective of combining reality and general applicability.

In Chapter 6, we present the experimental design and results of our benchmark study. The goal is to answer research subquestions 5, 6, 7, and 8. *Does the profit-driven sort insertion heuristic contribute to the quality of the solutions in the multi-start heuristic? What local search operators contribute the most to the objective function? What is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping? Under what circumstances is the use of transhipment hubs effective?* We start with a general introduction to the benchmark study, and present its design and parameter configuration. Subsequently, we report on the results of the experiments. We analyze and give an interpretation of the results, and compare our best network with the actual network that was used to create the data set.

Chapter 7 provides more in-depth coverage of the performance of the multi-start local search algorithm using several additional experiments. To assess the performance of our algorithm, we present an experiment that benchmarks the algorithm with heuristics from previous studies. Furthermore, we present an additional experiment that justifies the choice for a specific implementation of the transhipment operator. In addition, we discuss the influence of ship size, the initialization of multi-start parameters, and re-applying local search operators.

In Chapter 8, we present the conclusions of this research. We will shortly review the research subquestions that are answered in the previous chapters, and answer the main research question, *what is the performance of multi-start local search heuristics for solving routing and scheduling problems in liner shipping?* Furthermore, we propose suggestions for future research.

Finally, in Chapter 9, we provide some additional discussion on the experimental design and findings of this research.

Chapter 2

Liner shipping

In this chapter we will explore the problem domain and existing solutions. More specifically, we try to answer the first two research subquestions:

1. *What are the specifics of liner shipping and what are its typical problems?*

2. *What are multi-start and local search techniques, and why do they work?*

Section 2.1 gives a general introduction to liner shipping and sea transport. We discuss the history of liner shipping and containerization, present a typical service network, and elaborate on liner shipping economics and trends in the field. The problem we focus on, the routing and scheduling problem, is explained into more detail in section 2.2. The section also provides an overview of related research and existing solutions, such as multi-start and local search techniques. Finally, in section 2.3 we cover some of the problems that relate to either the routing and scheduling problem, or liner shipping in general, and are important to the sea shipping industry. A review on similar work from different sectors of transport is also part of the section's focus.

2.1 Introduction to liner shipping

As previously stated in the Introduction, the characteristic of liner shipping is that services are provided according to a regular repetitive schedule, similar to a bus line. Liner shipping carriers have a service network, consisting of all the services they provide. An example of a single service is illustrated in Figure 2.1, that shows Mitsui O.S.K. Lines' AEX Service between Asia and Europe. In Table 2.1, the corresponding travel times and port rotation are given. The complete cycle is usually divided into two parts, describing the direction of the trip. For the Asia-Europe trade lane these are the westbound trip

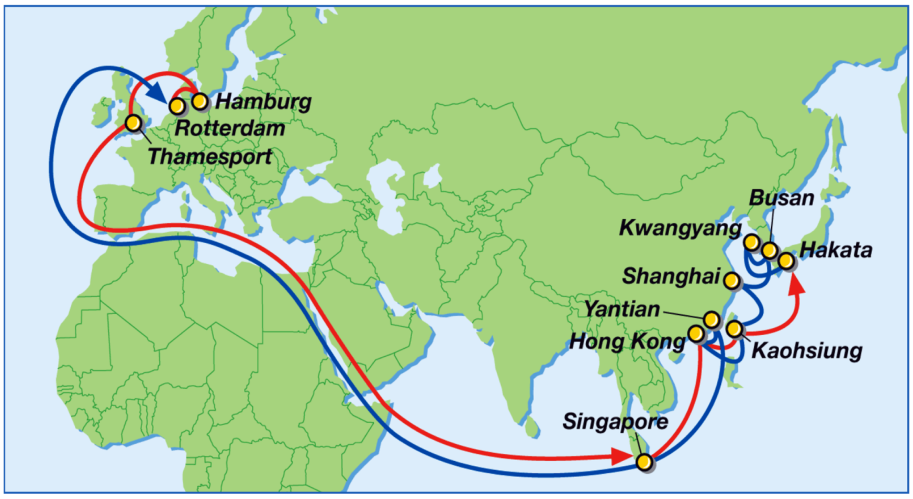


Figure 2.1: Mitsui O.S.K. Lines (MOL) - AEX Service.

Westbound	*Ports of discharge*		
Ports of loading	Rotterdam	Hamburg	Thamesport
Singapore	17	20	23
Yantian	21	24	27
Hong Kong	22	25	28
Kaohsiung	24	27	30
Shanghai	26	29	32
Busan	28	31	34
Kwangyang	29	32	35
Hakata	30	33	36

Eastbound	*Ports of loading*		
Ports of discharge	Thamesport	Hamburg	Rotterdam
Singapore	20	23	26
Hong Kong	25	28	31
Kaohsiung	26	29	32
Hakata	31	34	37
Kwangyang	32	35	38
Busan	33	36	39
Shanghai	35	38	41

Port	**ETA**	**ETD**
Hakata	Sat	Sun
Kwangyang	Sun	Mon
Busan	Mon	Tue
Shanghai	Wed	Thu
Kaohsiung	Fri	Sat
Hong Kong	Sun	Mon
Yantian	Mon	Mon
Singapore	Thu	Sat
Rotterdam	Tue	Thu
Hamburg	Thu	Sat
Thamesport	Sun	Tue
Singapore	Tue	Wed
Hong Kong	Sun	Mon
Kaohsiung	Mon	Thu
Hakata	Sat	Sun

Table 2.1: The left table shows the travel time between loading and discharging ports in days, for both the westbound and eastbound trip. The westbound trip is the blue line in Figure 2.1 that goes from Asia to Europe. The eastbound trip is the red line in Figure 2.1 that goes from Europe to Asia. The right table shows the port rotation with the estimated time of arrival (ETA) and estimated time of departure (ETD) for every port, specified as a day of the week. The AEX Service is a combined service of The New World Alliance partners. Source: Mitsui O.S.K. Lines (2010)

and the eastbound trip. On the westbound trip, cargo is loaded in Asia and delivered in Europe, whereas on the eastbound trip, cargo is loaded in Europe and delivered in Asia. The port rotation shows the *string of ports*, i.e., the sequence in which ports are visited to complete the cycle. Because ships usually sail along the coastlines, they will pass the ports that are situated on continents' edges in a specific order. We call this the *natural order*. Usually, ports in a string appear in their natural order. Sometimes, when two ports are close, a ship might deviate from the natural order. We can see this happening in Figure 2.1, where the ports Yantian and Hong Kong are exchanged. The same goes for the ports Kwangyang, Busan and Hakata, that are not visited in their natural order. In such cases, where ports are located near each other, minor advantages like slot allocation or terminal planning may compensate the extra sailing time.

Most services are provided on a weekly basis, that is, a ship visits the port of call each week on the same day. If a cycle takes n weeks to complete, this means there are n ships needed to provide the service.

Starting in the 1950s, maritime transport changed greatly with the adoption of steel containers, as a replacement of wooden boxes. However, it was until the late 1960s before the dimensions of containers were standardized, making an end to all incompatible container sizes. Liner shipping vessels carry only intermodal containers, of which the number of containers is measured in twenty-foot equivalent units (TEU), where one TEU refers to a 20-feet long container. Apart from the 20-feet container, the 40-feet and 45-feet containers are common sizes. Global container traffic has shown a substantial growth over the last decades, with world container port throughput increasing by 4 percent to 506 million TEUs in 2008 (UNCTAD 2009). The trend of increasing transport by containers is called *containerization*, and is a driving force of the globalization of international trade (Notteboom & Rodrigue 2008, De Souza Junior et al. 2003). The use of standardized containers provides for easy, cost efficient handling and storage in ports and on ships.

Over the last decades, liner shipping companies have significantly underperformed financially in comparison with other industries, due to its high capital intensity and the risk associated with the revenues (Notteboom 2004). Liner shipping involves higher fixed costs than, for example, tramp shipping since liner ships depart according to their schedule, regardless of whether the ship is fully loaded. Moreover, carriers have witnessed their profit margins decrease over time. The demand curves in liner shipping are rather inelastic (Notteboom 2004), leaving carriers with no choice than to accept low freight rates in order to increase capacity utilization and reduce their cost per TEU. To increase profitability, the companies focused on cost reduction and gaining economies of scale.

Since the mid-90s, we have seen large scale increases in vessel size in order to increase the capacity. This reduces costs since larger ships have lower cost per TEU-mile than ships with less capacity, albeit when the capacity is fully used. Todays largest vessels can hold over 10.000 TEU, the largest being the ships from Maersk's E-class (Emma Maersk, Estelle Maersk, Eugen Maersk, among others), that can hold over 11.000 TEU[1]. Cullinane et al. (1999) showed that economies of scale are gained starting from 8.000 TEU, at least for the Europe-Asia and trans-Pacific routes. Using larger ships does reduce variable cost in the long term, but the short-term capital investment is sky high. Moreover, the weekly departures of a liner service requires similar-sized ships. Upgrading a fleet involves high investment and takes several years.

The development of larger ship sizes does put restrictions on the service network the carrier plans to provide. These ships may not be able to use certain routes and visit every port. Ships larger than the maximum depth of canals or straits (e.g. Panamax, Suezmax and Malaccamax) are forced to use longer alternate routes. A similar constraint applies to ports, that have their own maximum draught. Liner ships load and offload their cargo in container terminals, which at the same time provide transhipment to other modes of transport to serve the hinterland. The handling of larger ships also has implications for container terminals storage space and the inland transportation system.

Another way to gain economies of scale is to co-operate with other carriers. Three forms of co-operation exist in liner shipping, namely, operating agreements, trade agreements, and mergers and acquisitions (Notteboom 2004). Operating agreements can be carriers sharing vessels or container slots, or forming a consortium or strategic alliance. A type of trade agreement is the conference, in which carriers control tariff rates in the market to prevent price wars. In mergers and acquisitions two carriers transform into one organisation.

The formation of global alliances in liner shipping started in 1996, in order to survive the severe competition the companies were in. This form of co-operation comes with several advantages, regarding to both planning and financial aspects. Carriers in an alliance maintain services together, such that each carrier needs less vessels to provide a

[1]Official number of TEU as specified by A.P. Möller - Maersk Group. However, when calculating the number of TEU to be carried independent of the weight of the containers, one would obtain a capacity of 15.200 TEU. Maersk assumes a weight of 14 tons per container for their calculations.

service. These extra vessels can be deployed on new lines, entering new markets. Another option is to keep the ships assigned to the line, and provide clients with more frequent shipping. In any way, the alliance extends the service network for both carriers, providing a wider geographical scope, assuming a different initial service network (Ferrari et al. 2008). Furthermore, alliances allow for sharing terminals. The financial advantages of alliances concern risk and investment sharing, and having more financial resources available to maintain and upgrade the fleet. Overall, the main advantage of global alliances can be specified as economies of scale. Despite the numerous advantages, the organizational complexity involved with alliances undermines trust among the partners, for example due to possible intra-alliance competition (Midoro & Pitto 2000). These difficulties are even greater with mergers and acquisitions (Fusillo 2009).

Today's most important alliances are the Grand Alliance (Hapag-Lloyd, NYK Line and OOCL), The New World Alliance (APL, Hyundai, MOL), and CKHY Alliance (Cosco, K-Line, Hanjin, Yang Ming). In 2006, the two biggest alliances, Grand Alliance and The New World Alliance, joined forces to withstand the giant Maersk Line after its acquisition of P&O Nedlloyd. The two alliances now provide several services together. The formation of alliances among carriers is a case of horizontal integration in the market. However, at the same time, the maritime industry experiences vertical integration with companies extending their control of the logistics chain by the same type of cooperations (Heaver et al. 2000). A good example is Maersk, that has developed almost door-to-door services by handling containers at their own terminals, and managing inland transportation.

Another important aspect of reducing costs nowadays is *bunkering* fuel oil, since the cost of fuel accounts for half of the total operating costs of a container ship (Ronen 2011). Bunker prices fluctuate constantly, such that the carriers want to bunker in the ports with the lowest price. The fiscal policies of the countries also influence this decision. Carriers have introduced surcharges to be able to pass increased bunker prices on to the customer, for example the Bunker Adjustment Factor (BAF). The BAF charges were determined for each route for a certain time period. These charges used to be set by carrier conferences, but in 2008 the European Commission banned these conferences, and now carriers set their rates individually. To reduce overall bunker costs, carriers focus on using cheaper grades of fuel, changing the vessel design, and travelling at slower speeds (Notteboom & Vernimmen 2009). The latter two aspects try to increase the fuel efficiency of the ship. Examples of changing vessel design are the use of newly shaped propellers, more efficient engines, and improved aerodynamics. Lowering the sailing speed reduces fuel consumption, but increases transit times and may put the carrier in a competitive disadvantage (Ronen 2011). An extra ship needs to be added to the service loop in order to maintain the shipping frequency. However, the savings achieved with lower speeds are considered to be marginal (Notteboom 2006). In that respect, carriers have to make a tradeoff between cost savings and schedule frequency. Over the last years, it turned out that most carriers decided to reduce service speed and add extra tonnage (Notteboom & Vernimmen 2009). This trend of *slow steaming* was realizable since the carriers faced overcapacity due to plumetting demand.

The global trend of increased environmental awareness has its impact on shipping as well. Although shipping is by far the most efficient way of transport when it comes to energy consumption per unit transported, and considerable cuts in CO_2 emissions have been made, reduction continues to be an important objective. In 1970, ships used 0.200 kg of fuel to transport one container one nautical mile on the trip from Asia to Europe, whereas in 2007 this figure has decreased to 0.025 kg of fuel (Christiansen et al. 2009). Shipping companies address the issue by explicitly stating objectives regarding the CO_2/TEU emission per nautical mile. Moreover, some local regulations oblige carriers to travel at slow speed in order to cut emissions (Morton 2007).

2.2 Routing and scheduling problem

Transport by sea is a widely discussed topic in literature. Although each type of sea shipping has its specific problems, the scheduling and routing of a fleet is a common issue. Attempts to find good solutions have resulted in numerous approaches. Roughly, we can distinguish between mathematical programming approaches and heuristic approaches.

2.2.1 Mathematical programming approaches

Mathematical programming approaches include (mixed) integer programming, dynamic programming, and Lagrangian relaxation. These approaches focus on formulating the problem, or several sub-problems, in a formal mathematical way after which the problem can be solved to optimality using optimization software.

In sea shipping problems, a well-known representation is the *multi-commodity flow problem* (MCFP). The MCFP is a special case of linear programming which considers the flow of different goods with various origin and destination nodes in an underlying network. Alvarez (2009) solves a mixed integer programming (MIP) formulation by formulating a MCFP, which is then solved using Simplex or an IP engine. Agarwal &

Ergun (2008) use a reversed approach. First, the MCFP is formulated using a directed, weighted graph. Then a MIP is formulated and solved using three different heuristic algorithms: a simple greedy algorithm, a column generation-based algorithm, and a Benders decomposition-based algorithm. Fagerholt (2004) considers a sub-problem of liner shipping: feeder lines. Here, the routing problem is deduced to a multi-trip vehicle routing problem (VRP). The feasible routes are used as input to an integer programming model. This IP model is solved using GAMS/CPLEX.

An alternative technique is proposed by Yan et al. (2009). To deal with the complexity of real-world applications, one of the constraints of the original IP model is *relaxed* with the corresponding non-negative Lagrangian multipliers and is added to the objective function. The relaxation of constraints allows the problem to be divided into several independent sub-problems, which are relatively easily solved. The adjusted IP model is then divided into a ship-flow network and a container-flow network. Finally, these two sub-problems are solved using CPLEX. Ting & Tzeng (2003) use dynamic programming (DP) to deal with the mathematical complexity. Each stage of the DP represents a candidate port of the planned route, and has some factors associated with the next stage, including crane productivity and cruising speed. The solving algorithm searches the optimal schedule by moving through the stages.

2.2.2 Heuristic approaches

Due to the high complexity involved with optimization problems, mathematical programming approaches may have difficulties finding the optimal solution in a reasonable amount of time. Most often, they can only solve small-sized data instances to optimality. Although relaxation techniques can transform optimization problems into related, but more easily solvable problems, the solution is only an approximation of the original problem's solution.

Heuristics potentially can provide near-optimal solutions in less amount of time. In the context of liner shipping, heuristic-based approaches include heuristic algorithms and metaheuristics like genetic algorithms.

Man (2007) proposes a variant of the set covering problem. The problem is adjusted to be applicable to liner shipping. First, the traveling salesman problem is applied to all possible port combinations. Then, a starting set of routes that covers all demand is selected by solving the adjusted set covering problem. In the final stage, demands are allocated to routes, taking into account the vessels' capacity constraints. The same approach is used by Van de Weerd (2009), who applies the adjusted set covering problem only to regions without mutual trade. The regions with mutual trade are handled by means of an approximative method.

Van der Meer (2009) proposes randomized optimization. In an iterative approach, all possible routes that (partially) satisfy demand between ports are generated and a profitable route is selected. This route is then added to the solution set only if it increases total profit, after which the demand matrix is updated and the loop restarts. This process repeats until the last 50 iterations yield no improvement in terms of total profit.

The complexity of the liner shipping industry relates primarily to the design of a service network. This service network constitutes of several routes, but simply joining the most profitable routes into a network will not yield the best combination, nor the optimal solution. After all, picking routes is a cascading activity, which means that choosing one route will change the type and order of the routes that follow. Such hard combinatorial optimization problems tend to get stuck in a local optimum. To find global optimal solutions, some kind of diversification is required. One way to achieve this is to restart the procedure from a new solution once a region has been explored (Marti 2003). Restart mechanisms have also successfully been applied to simulation optimization problems (Nicolai & Dekker 2009).

In sea shipping, the restart (or *multi-start*) heuristic has shown to generate viable solutions. Ronen (1986) compares three algorithms to solve relatively simple industrial shipping problems. The first algorithm is based on a generalized transportation problem and is mathematically solved. Although this procedure assures an optimal solution, the authors acknowledge that efficiency is a problem for larger-sized problems. The second algorithm is the 'biased random generator algorithm' and works by choosing ships at random, and assigning unloading ports at random, until the ship is full or the maximum number of unloading ports for the ship has been reached. The process is repeated to obtain many solution candidates. The algorithm is not 100% random, but *biased* because a shortest route calculation is performed for each ship. Finally, the third algorithm tries to minimize the cost per ton-mile of cargo. Although the approach worked quite well, in general it was the biased random algorithm that outperformed, and even found the optimal solution in 5 out of 8 small-sized problems.

In TurboRouter, a decision-support system, restart heuristics have been used in combination with local optimization techniques to solve the routing and scheduling problem in industrial and tramp shipping (Fagerholt & Lindstand 2007). The multi-start local search heuristic is more extensively discussed by Brønmo et al. (2007). Its approach is two-fold. In the first phase, a multi-start insertion heuristic is applied. This heuristic assigns a random percentage of the list of unassigned cargoes to randomly chosen ships of the (existing) fleet, after which these cargoes are removed from the list. The remainder of the list is sorted increasingly by the start of the pickup time window (*time sort*) or decreasingly by the cargo quantity (*quantity sort*). Next, all unassigned cargoes are iteratively assigned to the available ship that gives the highest contribution to the selected objective function. The complete process is repeated many times to generate a large number of different initial solutions (e.g., 1,000 times). One half is generated using time sort, while the other half is generated using quantity sort. From these initial solutions, a randomly chosen fraction of the best solutions is selected and passed on to the *local search heuristic*. In this second phase, local search operators are used in an attempt to improve the solutions. *Intra-route* operators try to improve the allocation of demand over a single ship, while *inter-route* operators look for improvements by moving cargoes between multiple shipping routes.

An extension of the TurboRouter algorithm is proposed by Brønmo & Løkketangen (2007). This study introduces an extra stage in the middle. The set of all the generated routes is first passed to a set partitioning algorithm. The general approach is to define a set of candidate routes, initially equal to the set of all routes. Then a single solution is selected by evaluating all candidates. To improve the quality of the solutions, the route evaluation is based on an adaptive, learning method. By selecting one route, all other routes for the given ship, and all routes that contain at least one of the cargoes in the chosen routes, are not feasible anymore and are removed from the candidate list. This process is repeated until the candidate set is empty, and a subset of routes is selected. The selection is then passed on to the local search heuristic. Results indicate that the extended algorithm does not outperform the normal algorithm in most of the cases, and considering the increased computational time this is not very convincing.

2.2.3 Routing and scheduling in reality

The previous two subsections covered scientific approaches for solving the routing and scheduling problem in liner shipping. However, in reality, carriers do not always use the scientific knowledge. Every liner shipping company has a planning department, with route managers responsible for one or several services. Most route managers perform ship routing and scheduling manually based on their professional knowledge and experience (Lam 2009).

In liner shipping, and other sectors like aviation, the carriers assign profits to single services. Carriers determine the profit of a particular service by decomposing the total profit gained with transportations that used the service. Note that it is necessary to compensate for the feedering services that move cargo from its origin to a port along the service route, and from a port along the service route to its final destination. After decomposition, only the profits belonging to the liner service are considered.

The profit is one of the performance indicators that route managers have in their toolbox. Another important measure is the utilization of capacity along the service. With these indicators, a route manager can adjust the service, for example by deciding to enlarge or reduce capacity on the route, and add or remove a port visit from the route. However, since carriers have route managers optimizing single routes, it is not clear what happens when focusing on a whole service network. It is very likely that a full optimization approach, i.e., considering a whole service network instead of single routes, as we propose in this work, would yield a service network that is more cost-effective. Furthermore, a big advantage of a full optimization approach is that it is more robust to changes. With the occurrence of events that influence liner operations, like plummeting demand or piracy, a full optimization approach allows for an easier revision of services, making the carrier's service network more resilient.

2.3 Related problems

In the following subsections we will discuss problems that are related to the routing and scheduling problem and to liner shipping in general. Although we do not provide an exhaustive overview, some problems will be briefly covered because they are important to the industry.

2.3.1 Other routing problems

The routing and scheduling problem is not only preserved to liner shipping, but occurs in other sectors as well. Apart from transportation by ship, these problems are common in transportation by truck, train (Cordeau et al. 1998), and airplane (Desaulniers et al. 1997). Most work has been focusing on the vehicle routing problem, in which a fleet of vehicles (trucks) delivers items to customers. The vehicle routing problem, proposed by Dantzig & Ramser (1959), is considered as one of the most important problems in logistics and transportation. The vehicle routing problem itself is a generalization of the traveling salesman problem (Potvin 2009), that focuses on finding the shortest tour for visiting a given set of cities. The traveling salesman problem is often considered as a subproblem in ship routing and scheduling.

As for many problems, the vehicle routing problem comes in several forms. The characteristics of the problem determine the specific variant considered. Examples of characteristics are the size and type of fleet, location of demands, the type of operation (pickup, delivery, or both), and possible time restrictions (Bodin et al. 1983). In addition, several objectives can be defined, ranging from minimizing costs and maximizing profit, to minimizing the number of vehicles required, and maximizing utility. Three common variants are the standard Vehicle Routing Problem (VRP), the Vehicle Routing Problem with Time Windows (VRPTW), and the Pickup and Delivery Problem with Time Windows (PDPTW).

In the standard VRP, a fleet of vehicles is located at a depot from which several customers have to be served. Each customer has a given demand and each vehicle has a given capacity. The objective is to minimize the distance traveled. The VRPTW differs from the VRP in the sense that the customers have to be served within a specified time frame. In the pickup and delivery problem, customers have transportation requests to be carried out by the vehicles. Each item to be transported has a particular size, which lays restrictions on the number of items that can be transported at once. Here, the objective is to find the optimal routes to fulfill all requests. In the PDPTW, the requests have to be fulfilled within a specified time frame. Several solution approaches exist for these problems, ranging from mathematical programming approaches (Toth & Vigo 2001), to metaheuristics (Bräysy & Gendreau 2005), and heuristics (Laporte et al. 2000).

Sometimes vehicle routing is considered in conjunction with other problems along the supply chain, like planning and scheduling. In most cases, the specific application requires such an integrated approach. A nice example in this perspective is the work presented in Naso et al. (2007). Here, the authors consider the production and just-in-time delivery of ready-mixed concrete. A metaheuristic solution approach is presented, combining a genetic algorithm with constructive heuristics. Naturally, the transportation of ready-mixed concrete is constrained to strict time limits, making the problem even more complex than standard delivery problems.

The field of shipping has received relatively little attention. The most important reason is that ship routing and scheduling problems exist in a much larger variety with respect to the problem structure and operating environments (Ronen 1983). This is already illustrated by the fact that we distinguish three types of operation, namely industrial shipping, tramp shipping, and liner shipping. Ship routing and scheduling differs from the vehicle routing problem in the sense that ships operate under different conditions (Christiansen et al. 2004). For example, in shipping, a fleet consists out of heterogeneous vessels with different capacities, speeds, and cost structure (Ronen 1983).

Furthermore, ships do not have to return to their origin, and since multiple crews are on board, they can be operated around the clock. In addition, the voyage of a ship takes days or weeks to complete, and ships may face port operation time windows. For a more extensive comparison of operational characteristics among freight transportation modes we refer to Christiansen et al. (2004).

2.3.2 Fleet sizing problem

Shipping lines operate a number of vessels that are deployed on specific shipping routes. To ensure a weekly service, carriers need a lot of vessels, usually around five to ten vessels for intercontinental routes. Fleet sizing involves controlling the number and size of the vessels used (Dong & Song 2009). Obviously there is a tradeoff involved: a smaller fleet size saves capital costs but may increase risk of loosing customer demands due to unavailability.

A three-phase solution method is proposed by Fagerholt (1999). In phase 1, all feasible single routes are generated for the largest vessel available. A ship's capacity utilization is used when calculating the cost of each route. In phase 2, single routes are combined into subsequent series, first into double routes, then into triple routes, and so on, each time determining the minimum ship size required to perform the combined routes. Because routes are subsequently considered, the minimum ship size of a combined route can easily be determined by selecting the largest minimum ship size across all single routes. In stage 3, a set partitioning problem is solved to find the optimal fleet.

We have seen the size of container ships grow steadily. While ship capacity was limited to 5,000 TEU in the early 90s, an increasing portion of todays ships have capacities over 10,000 TEU ('mega-ships') or even 12,500 TEU ('Ultra Large Container Ship', ULCS) (Imai et al. 2006). Larger ships typically have a lower cost per TEU-mile than smaller vessels with the same load factor, primarily because they are more fuel-economic. However, the economies of scale are limited. Larger capacity will generally lead to poorer capacity utilization, and the need to buy more cargo at lower rates. There are strong indications that the range of 5,500 to 6,500 TEU will reveal to be the most competitive vessel size (Notteboom 2004).

2.3.3 Master bay planning problem

The time a ship spends in a terminal consists mainly of loading/unloading activities and waiting time before and afterwards. Container loading is the most complex task and also the key variable determining terminal efficiency. Its complexity has to do with the number of constraints. There should be equal weight distribution in all directions (left/right and front/rear), heavy containers should be lower than light-weight containers, containers with an earlier destination should be easily reachable and grouped, (loading) crane utilization should be optimised, etc.

The stowage layout of a ship is called the *Master Bay Plan*. Such plans are usually designed on a daily basis by terminal management. Ambrosino et al. (2009) propose a heuristic approach to solve the Master Bay Plan Problem. In the first phase, bays are assigned to containers according to their destinations. Next, a trial solution is generated by solving the individual single-destination stowage problems, disregarding part of the constraints. Finally, in phase 3 a tabu-search meta-heuristic is used to determine a feasible solution, i.e., with respect to all constraints. The study also presents a software application using the heuristic that can serve as decision support system.

2.3.4 Empty container repositioning

Empty containers are usually stored in ports or repositioned to other ports to meet demands there. Repositioning empty containers from one port to another is also an essential mechanism to overcome the trade imbalance arising from most of the geographical locations (Dong & Song 2009). It was reported that empty containers have accounted for at least 20% of global port handling activity ever since 1998 (Drewry Shipping Consultants 2007). The drawback, of course, is that transporting empty containers occupies container slots on a vessel, resulting in lost opportunities to yield freight revenues for those slots.

Shintani et al. (2007) recognize the problem of empty container repositioning and propose a service network design with incorporated spare space on ships to enable repositioning. Furthermore, containers are leased when empty containers do not arrive at the demand points in time. Genetic algorithms are used to determine a solution.

Feng & Chang (2008) formulate empty container repositioning as a transportation problem. The objective is to minimize the costs of empty container repositioning. The first step identifies ports that have a surplus of empty containers, and ports that have a shortage. The second step exploits Liner Programming to solve the transportation problem.

2.3.5 Transhipment hub locations

Ever since the early days of containerization the shipping industry is searching for hub locations to limit the number of port calls in one route. In the airline business, it is very common that large airplanes only call some of the main airports of a country, and that smaller airplanes (feeder services) provide onward connections. However, the transhipment of sea containers is relatively more costly, resulting in intercontinental container carriers that still call multiple ports at each end of a route. Furthermore, the choice of a transhipment hub is less obvious.

Although some transhipment hubs are used naturally because of their geographical position, carriers tend to have different ideas as to which is the best hub port to concentrate transhipment flows for a given region.

Baird (2006) explores the costs of current major hub locations in Northern Europe compared to alternative, new locations in that region. With the help of an extensive cost model, they conclude that, by average, cost savings up to 10% could be obtained when choosing a new hub location.

Aversa et al. (2005) introduce a mixed integer programming model to determine optimal hub port locations. The model considers 11 feeder ports at the East Coast of South America as hub candidates. The study reveals that port costs are a major decision variable, which subsequently explains the development of carriers' dedicated terminal facilities.

2.4 Conclusion

In this chapter, we have introduced liner shipping and identified its main problem. At this stage, we are able to answer the first two subquestions.

1. *What are the specifics of liner shipping and what are its typical routing and scheduling problems?*

Liner shipping is characterized by carriers providing a regular repetitive schedule of services, i.e., ports are visited weekly. Since the routes are fixed for longer time periods, liner shipping is less flexible than other types of shipping. The services a carrier provides together form the carrier's service network. An important problem in liner shipping is determining a service network of routes and frequencies that, for more or less given demand between ports and a fleet's capacity, maximizes profit. We refer to this problem as the routing and scheduling problem in liner shipping. The solution approaches in ship routing and scheduling research that we distinguish are mathematical programming and the use of heuristics. Mathematical programming approaches yield the optimal solution, but require substantial computation time. Heuristic approaches do not guarantee to find the optimal solution, but at least provide a near-optimal solution in a short amount of time. In practice, these scientific approaches are not very common. Instead, carriers have route managers that are responsible for single services, who can alter the routes or capacities based on performance indicators like utilization and profit. Although such an approach works, we focus on the optimization of a full service network since it is likely to be more cost-effective than altering single routes.

2. *What are multi-start and local search techniques, and why do they work?*

Creating a service network is a very complex task since picking routes is a cascading activity. This means that choosing one route will change the type and order of the routes that follow. Such hard combinatorial optimization problems tend to get stuck in a local optimum. In order to find the global optimum, some kind of diversification is required. This can be achieved using multi-start techniques. Multi-start refers to restarting an algorithm with different parameters, such that several regions of the solution space are explored. Local search is a heuristic optimization technique that tries to improve initial solutions by moving through the solution space and searching for neighboring solutions that maximize a given objective. The multi-start technique is very suitable for usage with local search, since it decreases the chance of ending up in a local optimum.

Chapter 3

Formulation of the routing and scheduling problem in liner shipping

In this chapter we define the precise variant of the problem we consider. As stated in the previous chapter, the routing and scheduling problem comes in different forms, even within the field of liner shipping. Problem variables can be considered or left out, and those considered can be assigned different values. The resulting set of problem variables and their values specify the precise variant of the problem. First, in section 3.1, we define the assumptions underlying the problem we consider. These assumptions help narrowing the scope of the problem, and contribute to the feasibility of the research. Then, in section 3.2, we give a more extensive problem description based on the assumptions. In addition, we discuss important problem variables like revenue and costs.

3.1 Assumptions

Numerous studies are dedicated to the routing and scheduling problem, each with their own scope and slightly different assumptions. A detailed definition of the assumptions helps scoping the research, and puts this research in perspective with the existing literature in the field. Furthermore, it provides a concise overview of the variables and issues that relate to the routing and scheduling problem. Overall, we distinguish between three types of assumptions, either representing reality, extending reality, or delimiting the scope of the study.

Assumptions representing reality
In order to apply the problem to liner shipping we need assumptions representing reality. It is important to state these assumptions to put this research into perspective, since today's reality may change over the years. The assumptions corresponding to this type are the following.

1. We consider liner shipping of TEU containers with multiple origins and routes, and multiple loading and discharging ports. The pick-up and delivery of containers is interwoven. We focus on a problem size consisting of around 50 ports.

2. For every route, there is a call once a week, at a fixed day. This assumption implies that if the duration of a trip is n days, $max(\frac{n}{7})$ vessels are needed on that route.

3. The demands between ports do not necessarily have to be satisfied. To enable profit maximization, we allow partial satisfaction.

4. Routes do not have to be symmetric, some ports may be visited only once on a route, while others may be visited twice, i.e., both on the westbound and eastbound trip.

5. Distances between ports are given.

6. The total costs involving sea transport can be divided into fixed costs and variable costs. Each vessel incurs capital and operating costs (e.g. the cost of leasing, insurance, maintenance and wages), which are fixed costs for one year of operation. Fuel costs depend on the distance travelled. Port costs include fees for entering, exiting, pilotage, and the accompanying towage, and are considered fixed for every port. In addition, the ports charge a fee per TEU loaded or offloaded.

7. The revenue per TEU is considered to be dependent of the direct distance between origin and destination of the cargo, as well as the direction traveled, and will be defined later.

Assumptions extending reality
Sometimes we need to extend reality in order to simplify the problem. The following assumptions belong to this type.

8. The fleet has an unrestricted size, and consists of vessels of similar size and capacity. As a consequence, the vessels have the same operating costs.

9. Each port is capable of receiving and handling a vessel of any size.

Assumptions delimiting the scope of the study
Assumptions delimiting the scope of the study prevent the problem from growing too complex. This is necessary to ensure the solvability of the problem, especially in the case of routing and scheduling in liner shipping, which relates to many sub-problems. The assumptions that belong to this type are the following.

10. The cruising speed of the vessels is considered constant and the same for all vessels in the fleet.
11. The demand between ports is deterministic and constant over time. The demand between ports is measured in annual TEU and is given.
12. Only interregional demand is served by liner shipping, intra-regional demand is fulfilled by means of local transport. This also means that feeder lines are beyond the scope of this research.
13. The time spent in a port is considered to be constant and independent of the number of TEU to be loaded or offloaded.
14. Ports are visited in the natural order of the string of ports. In Chapter 5, we provide an example based on our data set. Since ports lay along coastlines, and the continents have a convex structure, the ports occur in a natural order. This assumption avoids the complex Travelling Salesman Problem, which is normally considered in ship routing problems.
15. There are no time windows specified for the delivery of containers.
16. The slack of a route is considered for a whole round trip only, and should be between a minimum and a maximum. The slack of a route is defined as the time between the duration of a round trip and the next integer number of weeks. This means a minimum number of days is specified as additional buffer for one round trip. The maximum slack depends on the days left until the next cycle, and needs not to be specified.

3.2 Problem description

Given the yearly demand (in TEUs) between ports, the objective is to design a service network that maximizes profit. Let P denote the set of ports in the aforementioned regions. Demand is assumed to be deterministic, and is represented by a origin-destination demand matrix Q, where each of its elements q is associated with a specific origin port o and destination port d. Hence, $q = Q^{(o,d)}$. Obviously, $o, d \in P$.

A service network is defined as a set of active routes, where each route contains a sequence of ports calls. This sequence is called a *string*, also known as the *string of ports*. A string of ports can be described in the following format: 1-2-4-5-6-5-3. Since strings in liner shipping represent *cycles*, we do not allow the starting and end point to appear twice in the string. In this case, port 1 is the starting and end port, and port 6 is a turning point (i.e., referring to a change in direction). Strings do not have to be symmetric, and may contain ports that appear twice. We provide more details on the appearance of strings in Section 4.2. Designing a service network consists of creating routes and allocating the demand over these routes. Ships depart weekly from each port along the route to provide customers with a regular schedule.

The objective is to maximize the profit of a single carrier. We do not consider a competitive market with multiple carriers. We define profit as the total revenue minus total costs. A container generates revenue during transport, so the total revenue in one year equals the sum of all containers' revenues.

3.2.1 Costs

Deploying a service network incurs costs. We distinguish between different types of costs, based on the cost structure in Man (2007).

Capital costs

First, there are *capital costs* involved with the ownership or leasing contracts of the fleet. These costs mainly depend on the capacity of a vessel. In case of full ownership, capital costs consist of finance costs, such as interest and depreciation. When a fleet is insufficient, additional vessels might be leased, either for short-term or long-term. In that case, the capital costs are equal to the leasing price. We express capital costs as an annual sum, fixed for each type of vessel.

Each service network has some associated routes. For every route, there is a call once a week. This implies that if the duration of a trip is n days, $max(\frac{n}{7})$ vessels are needed on that route. So, there is a direct relationship between capital costs and the total number of sailing days in a service network. If a route is added to a network, or a route is extended, the impact on capital costs can be substantial when an additional vessel has to be used.

Operational costs

Operating a ship involves maintenance, insurance, wages, etc. We call this *operational costs*, which is a fixed amount.

Fuel costs

A large portion of the budget, however, is spent on *fuel costs.* Fuel costs depend mainly on the distance traveled, but other factors affect fuel costs as well. Fuel consumption for example is based on cruising speed, vessel draught, and ocean streams. Reducing the cruising speed can have great influence on the fuel cost per nautical mile. Fuel price is another cost driver, although in practice the fluctuations are somewhat leveled by surcharges to the customer, also known as the *bunker adjustment factor* (BAF). We choose to set a fixed cost per nautical mile, disregarding any other variable. This also simplifies the behavior of our algorithm. For example, extending a route with one port will increase the traveling distance, and therefore the fuel costs.

Port costs

Ports charge a fee for entering, exiting, towage, etc. These *port costs* are independent of the number of containers and are considered to be fixed in this study, although in practice port costs depend on vessel capacity, length, draft, etc. Normally, port costs also vary per port, since these costs are often used as as competitive instrument. In this study, we assume the fixed terminal costs are included in the port costs.

Handling costs

Finally, loading and offloading containers incurs *handling costs*, a fixed sum per TEU, charged by the terminal. This is a crucial variable when considering transhipment, because of the high extra costs involved with (off)loading containers.

3.2.2 Revenue

The revenue originates from containers transferred from their origin to their destination. Although some studies, for the sake of simplicity, assume fixed revenues per container, real life examples like public data from Maersk show that the price of a container service actually depends on the distance it travels. So, the further a container has to travel, the more revenue it generates. This seems sensible, because the costs per mile of a container mainly consist out of fuel consumption. The height of the price also tends to depend on the direction of the transport. For example, on the Asia-Europe trade lane, eastbound transport is less expensive than westbound transport. In practice, the freight rates on the two directions are also rather asymmetric.

It is common practice to express the price, and therefore the revenue of a container service, in dollars per nautical mile or, in some cases, per sailing day. We will adopt this approach and assume a fixed revenue per TEU per nautical mile. This actually creates a pitfall that we have to be careful about. Assuming that revenue increases with the distance traveled, this also means that we can make more money by keeping the cargo on board for a longer period. However, that is not realistic. Strong competition places prices under huge pressure, leaving carriers with no other option than to offer competitive prices and transit times. In other words, a carrier cannot artificially boost prices. To account for this problem, we use a distance *norm* in our revenue calculations. Suppose we call port A, B and C, and we have to take a container from A to C. In this example, we use the direct distance between A and C to calculate the revenue from that container, in stead of using the distance between A and B plus the distance between B and C.

The result is a fair method to calculate revenue. Also, it forces the carrier to choose cost-efficient routes. For example, if more round trips call the origin and destination port of a demand, the carrier is forced to choose the route that has the shortest path between origin and destination to reduce his costs. While this reasoning might seem obvious, we have to account for it in our algorithms. In Chapter 4 we will provide more details on the algorithmic consequences.

3.2.3 Consequences of chosen cost-revenue structure

The chosen cost structure, and to some extent also the revenue structure, influences the composition of a service network. Each cost parameter has its own effect on the network. For example, by defining port costs to be independent of the number of containers, or ship size, we enlarge the ‘economies of scale’ effect, i.e., the savings that can be obtained by using larger ships. Hence, we expect that a network will try to use large ships rather than small ships. Similar remarks can be made when reviewing fuel costs. In practice, fuel consumption depends on the sailing speed. When fuel prices are rising, carriers tend to decrease sailing speed to save fuel costs. But that will increase a cycle's duration, urging for larger fleets.

Besides the structure, the values we choose to assign to these variables are also very important. For example, higher handling costs will certainly diminish the 'value added' of the transhipment operator, since the use of hubs becomes more expensive.

Given particular settings for cost and revenue parameters, we can identify a simple rule. A carrier needs a minimum volume before it becomes profitable to service a new port. Suppose we have an existing service between East Asia and Western Europe, and we would like to add Jebel Ali because we have some demand from that port. Although Jebel Ali appears to be 'on the route', we would still have to sail additional miles on each round trip to call that port. For example, the direct distance from Singapore to Rotterdam is 8,255 nm. However, the distance Singapore - Jebel Ali - Rotterdam totals 9,532 nm. So, each round trip we face 1,277 additional miles. This incurs additional port costs and fuel costs. Each container will generate revenue, but also incur handling costs. We can now estimate the minimum required volume, based on the cost and revenue surpluses. We assume port costs to be \$25,000 per entry and fuel costs to be \$150 per nautical mile. Furthermore, we assume that the average revenue per TEU from and to Jebel Ali is equal to \$475, while the handling costs are \$175 per TEU. The rationale of these values are outlined in Section 6.2.1. The calculation is shown below in Example 3.1.

Let d denote the distance, p port costs, f the fuel costs, c the number of TEU.

$$\begin{aligned}
d &= 52 \cdot 1,277 = 66,404 \text{ nm} \\
p &= 52 \cdot \$25,000 = \$1,300,000 \\
f &= \$150 \cdot d = \$150 \cdot 66,404 = \$9,960,600 \\
revenue &= (475 - 175) \cdot c = 300c
\end{aligned}$$

We find the minimum number of TEU when total costs equal total revenue:

$$\begin{aligned}
300c &= p + f = \$11,260,600 \\
c &\approx 37,535 \text{ TEU}
\end{aligned} \tag{3.1}$$

The previous example shows that the cost structure forces the carrier to evaluate the network composition carefully. Because of the costs involved with a weekly service, a minimum volume from and to each port is necessary to ensure profitability. The rough calculation above shows that at least 37,535 TEU are required to cover for the additional costs involved with a weekly service to Jebel Ali. The impact on both the ship capacity and the cycle duration can also have financial drawbacks. This however, entirely depends on the structure of a network, and is beyond the scope of this example.

More generally, if a carrier wants to include a port in its weekly service, he should make sure that at least the port costs are covered by the revenue. The revenue depends on the distance between origin and destination of a cargo, so that we cannot use a generic rule to indicate the minimum number of containers to compensate for the ports costs. However, following the parameters mentioned above, if we use an average revenue rate of \$475 per TEU we can calculate the required volume. From Example 3.2, it follows that any volume below 4,333 TEU implies that a port will be unprofitable. Again, this figure can be totally different when a container generates a lower or higher revenue due to the origin-destination distance.

$$\begin{aligned} p &= 52 \cdot \$25,000 = \$1,300,000 \\ 300c &= p = \$1,300,000 \\ c &\approx 4,333 \text{ TEU} \end{aligned} \tag{3.2}$$

3.2.4 Concluding remarks

We can draw the following conclusions from the chosen cost structure. Naturally, the carrier wants to maximize revenue and minimize costs. Capital costs can be regarded as sunk costs, since these are expressed as an annual sum. Once this sum is paid, the objective becomes to maximize the revenue per nautical mile. The carrier can do that by maximizing the utility rate of its fleet, and by entering long-distance cargo contracts. Obviously, the carrier tries to minimize the costs per mile as well, but the majority of those costs are incurred by fuel consumption. Since we consider sailing speed to be constant, the carrier has only little influence on the costs per mile. Port costs depend on the number of port visits, but these costs appear to be rather marginal compared to the fuel costs. Therefore, the carrier would like to make sure it can transport enough containers to and from a certain port to cover for the fuel, port, and handling costs.

Chapter 4

Multi-start local search heuristics

In this chapter we present a multi-start local search algorithm for the routing and scheduling problem in liner shipping. In multi-start local search heuristics, different initial solutions are generated in the first phase, and improved by local search in the second phase. The research subquestion we try to answer is:

3. *Can we transform the multi-start local search algorithm from tramp shipping studies to the case of liner shipping?*

We start with a general overview of the two-phased model, followed by a detailed description of the individual steps.

4.1 Introduction

The complexity of designing a service network can be traced back to the difficulty of picking the routes that *together* form the most profitable network. Choosing one route will change the type and order of the routes that follow. Therefore, the profitability of the optimal network can never be calculated up front, without trying all route combinations first. Since this 'enumeration approach' is practically infeasible for real-life examples, alternative approaches have been designed recently. One of the most successful heuristic approaches to overcome the hard combinatorial optimization problem is the *multi-start heuristic* (Marti 2003, Nicolai & Dekker 2009). Applied to liner shipping problems, we often find multi-start heuristics combined with local search optimization (see Fagerholt & Lindstand 2007, Brønmo et al. 2007, Brønmo & Løkketangen 2007). The basic idea is to construct many initial solutions throughout the solution space, after which each solution can be subjected to a local optimization process. This approach provides an effective way to explore the solution space, without the requirement to iterate through each and every possible route-combination. Although optimality is not always reached, with carefully chosen parameters this heuristic approach can approximate optimal solutions.

The multi-start local search algorithm that we propose is two-fold. The first phase is the *insertion heuristic*, which generates a large number of different solution candidates. This phase works by randomly creating routes and assigning partial demand to them, similar to the technique used by Ronen (1986). The rest of the demand is then sorted and allocated to the routes, which together form one service network. This process is repeated to generate multiple, diverse solutions. In the second phase, local search operators are applied to improve each solution. Figure 4.1 shows the individual steps of the algorithm.

In the following sections we will comment on the details of the insertion and local search heuristic. In Section 4.2 we will introduce the insertion heuristic. Sections 4.2.1 and 4.2.2 describe the sorting methods we apply to obtain qualitative initial solutions. The second phase, local search optimization, is described in Section 4.3.

4.2 Phase 1: Generating initial solutions

The diversity of the initial solutions is very important for the performance of the multi-start local search algorithm. The local search heuristic will be used to explore the local neighborhood of each initial solution in an attempt to find its local optimum. Only when the set of initial solutions is spread throughout the solution space, we will have a good chance that one of the local optima equals the global optimum.

We use the *biased random algorithm* from Ronen (1986) to generate part of the initial solutions. The prefix *biased* refers to the prespecified order in which ports must appear. Initial solutions therefore are not perfectly random. We use only a small fraction of the cargoes to be randomly assigned. Then, we improve the quality by sorting the remainder of the cargoes and assigning them to routes in a way that it contributes to a specific objective function the most. We will focus more on the sorting functions in the next subsections. The biased random algorithm works by randomly generating routes and assigning randomly chosen cargoes to them. Each route consists of a series of ports that is cyclically called by a ship.

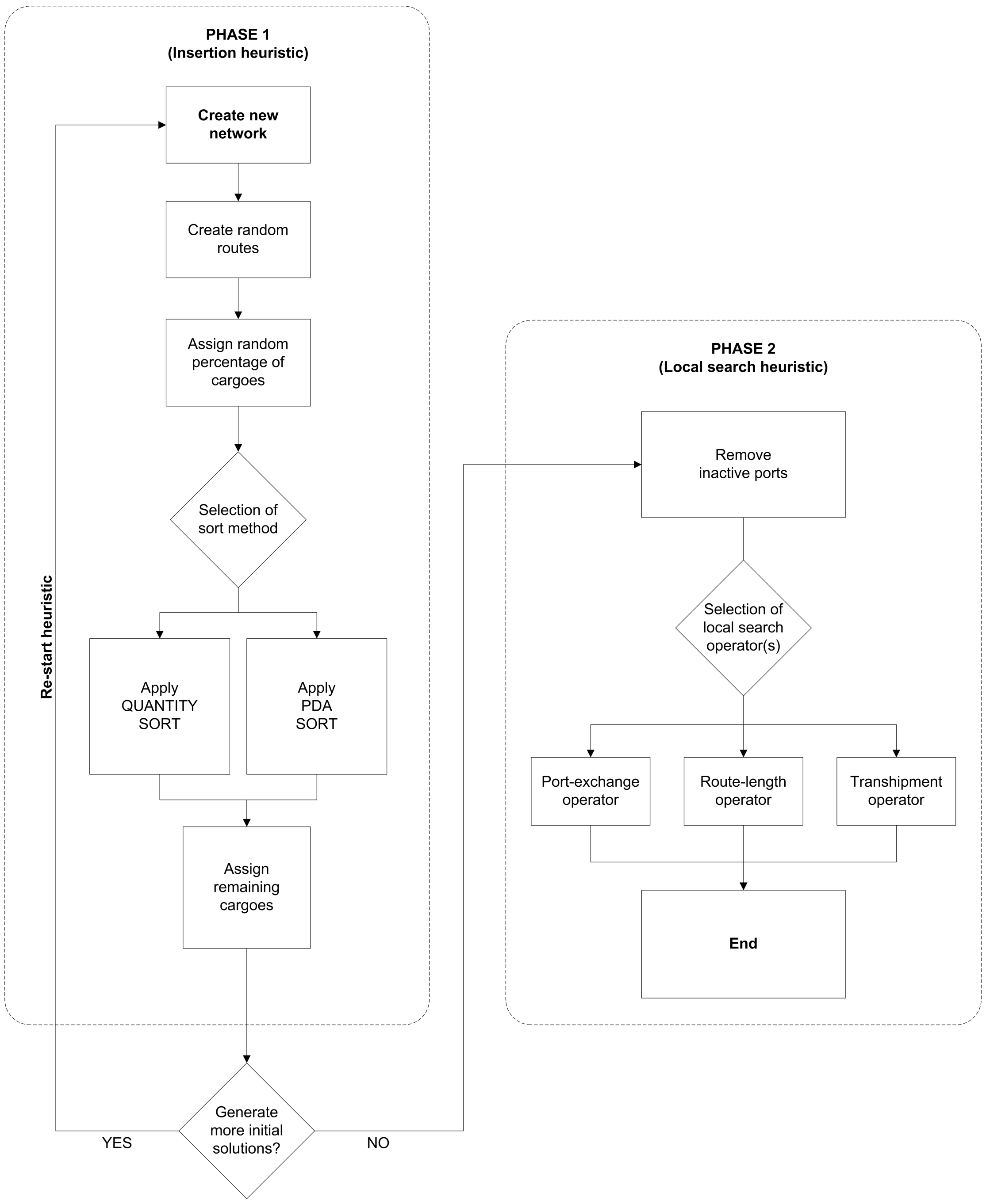


Figure 4.1: Outline of multi-start local search algorithm

String generation

In the first step we identify all the ports that have demand, and randomly select a subset. Because we have a round trip, we allow duplicate ports in a subset. The subset is then ordered to match its natural sequence. All ports are identified by integers, such that ports which are geographically succeeding, also have subsequent integers attached to them. This ordering procedure prevents the use of applying the Traveling Salesman Problem (TSP) to series of ports. Also, the lowest integers should resemble ports on one side of the trade lane, while the highest integers are the ports on the other side. For example, in case of an Asia-Europe service network, the lowest integers would represent ports in Asia, while the highest integers represent ports in Europe. The other way around would actually work just as well, as long as we make sure that ports in the middle of the trade lane (e.g., the Middle East) are identified by the integers in between.

Since the first and last port in a cycle are turning points, we prevent duplicates of those ports to be in the subset. The size of the subset is unconstrained, that is, within rational bounds: the minimum size of a subset is 2 because a round trip of only 1 port makes no sense, and a subset can never be larger than twice the number of existing ports minus 2, because we prevent the first and last port (turning points) to appear twice.

Ordering the string

Now we have to order the string in a sensible way. We numbered ports according to their natural sequence, so ordering should not be very difficult. We start with two positions that are already known: the first position and the middle position (the two turning points) are the lowest and highest integer respectively. In order to improve the understanding of string construction, we introduce terms to differentiate between the directions of the trip. Following our previous example, the part between the first position and middle position is called the westbound part of the string, and the part after the middle position is called the eastbound part of the string. Naturally, these directional terms are solely used in the text, and do not influence the applicability of the algorithm.

Out of the remaining integers, we pick the duplicates and divide each duplicate pair between the eastbound and westbound part. The remaining integers are the ones that can be assigned to either the eastbound or westbound part. This assignment is done randomly to avoid bias. Both eastbound and westbound parts are then sorted ascending and descending, respectively, to match the natural sequence. The final string now contains a perfectly ordered round trip.

Let us provide a small example. Suppose we have 10 ports numbered 1, 2....10, and suppose we have randomly chosen the following subset: 2-3-4-4-5-5-6-8 (here it is represented ordered for visualisation purposes only). In the first step, we pick the starting point and middle point, obtaining: 2....8.... (the dots have no meaning other than spacing). Next we pick the duplicates and divide each pair to the eastbound and westbound part: 2-4-5-8-4-5. The only remaining integers now are the singles ones: 3 and 6. Suppose that 3 is randomly assigned to the eastbound part, while 6 is assigned to the westbound part. The string then becomes: 2-4-5-3-8-4-5-6. After ordering, the final string is 2-3-4-5-8-6-5-4.

Partial cargo allocation

Now that we have a round trip we can start assigning cargo (demand) to it. The demand matrix we have is an origin-destination matrix with aggregated, yearly demand between ports. Other than spot cargoes in tramp shipping, we assume that individual demand at ports is aggregated and can be partially fulfilled in order to maximize profit.

As a first step, we will choose an origin and destination port arbitrarily and allocate a percentage of the demand between those ports to the round trip we just created. Assigning random cargoes ensures diverse starting points in the solution space. Imagine that we generate many strings in the previous steps. It is likely that some of them will be equal, although they can be in different networks. An ordered allocation phase, as we will introduce in the following steps, will allocate the same demands to similar routes. With the partial cargo allocation step, we ensure that all solutions will start in a different part of the solution space.

We purposely assign a small fraction (uniformly distributed between 5% and 15%) of the demand, because we would like to add quality to the solutions in a later stage by sorting the demands. This procedure has shown to improve solution quality in Brønmo et al. (2007).

After we selected the origin and destination port, and determined the fraction of the according demand, we check if the route has enough remaining capacity between those ports. If so, we allocate the demand fraction to the route and update both the demand matrix, and the remaining capacity of the route. If there is not enough remaining capacity, we pass this round. The procedure is repeated three times to insert random demands in each route.

At this point, we have a round trip that has some cargo between different pairs of ports allocated to it. For the moment, the round trip is ready. We can start adding new round trips, until we reach an predefined number of rounds.

Up to this point, we performed steps 1 to 11 of Algorithm 2. We now have a set of routes with some allocated cargo. In the following steps, we will sort the remaining demand. We choose one of the two possible sorting options, either quantity based, or profit-driven. One part of the solution candidates (service networks) will be sorted by quantity, the other part is sorted using a profit-driven criterion. By allocating the remaining cargoes in a sensible fashion to the routes, we already add quality to these initial solutions. The underlying idea is to improve solutions and speed up the local search process. In Sections 4.2.1 and 4.2.2 we provide more details on the sorting methods.

4.2.1 Quantity Sort

In the preceding steps we randomly allocated a small part of the demand to the routes in every service network. This ensured diverse starting points in the solution space. We now have to add further demand to the routes to allow viability of the networks. Although we could use random allocation for the rest of the demand, previous studies show that ordering methods improve solutions (Fagerholt & Lindstand 2007, Brønmo et al. 2007).

We start with a simple quantity based ordering method. In this approach, we start allocating the largest demands first. The underlying idea is that larger demands typically are more cost-effective, because port costs per TEU decrease with larger quantities. By assigning higher priority to larger demands, we make sure that we don't run out of ship capacity before the larger demands have taken care of.

The idea originates from a taxi-like scheduling problem in Madsen et al. (1995) and has been applied to the tramp shipping problem by Brønmo et al. (2007). Tramp shipping is characterized by long term cargo contracts and spot cargoes. As such, cargo demands can be individually distinguished and considered. Although we normally can identify individual cargo demands in liner shipping as well, our data set lacks this level of detail. We can only identify aggregated demand at each port, which means we will not be able to sort on cargo size at an individual level.

Instead, we will focus on the aggregated demand at ports. Since each port has a certain demand associated with it, we can use this information to assign a higher priority to the larger ports. We execute the following steps to allocate demand to the routes within a service network.

First, we sort the list of demands decreasingly by quantity. Then we start iterating through this ordered list. In each round, we find the routes that call the origin and destination port of a demand, which is obviously a rigid requirement. We then sort the list of feasible routes decreasingly by remaining capacity, because we want to allocate as much of the demand to one route as possible. Next, we walk through this ordered list of routes, each time allocating (part of) the demand until it has been completely allocated, or no more routes are available. Of course we update the list of demands as well as the remaining capacities of the routes throughout the process.

After this procedure, all demands have been allocated, unless some of them could (partially) not be allocated due to capacity constraints, or a lack of routes that call both the origin and destination ports. This Quantity Sort technique will be applied to the initial solutions with a probability $p = 0.5$. The other part is covered by PDA sort, which we will explain next.

4.2.2 PDA Sort

We use the term PDA as an abbreviation of 'profit-driven allocation', which refers to one of the many possible methods to allocate cargo to ships. This method works by allocating profitable cargoes first, before the remaining capacity of a ship is filled with less profitable cargoes. The underlying idea is that profitable routes together will also form a profitable service network. Although we explained earlier that the *best* service network is often not a composition of the best single routes, it seems intuitive that a service network will benefit from choosing profitable routes.

In our problem we face initial solutions that are improved by local search techniques. The best service network will then be the overall winner. It makes sense that the local search phase is more likely to find good, or even the best solution, when the initial solutions are already good.

We introduce a sorting method that is indirectly based on profitability. We sort the routes according to a three-attribute based scoring system that relates to profitability.

The first attribute is the length of a path, which is defined by the number of ports between the origin and destination. A shorter path will reduce the costs, because the distance is shorter. Although ports are ordered in their natural sequence, the direct distance from port A to port C is still shorter than the summed distances from A to B and from B to C. Tests have indicated that the average distance saved is between 5-10%.

The second attribute considers whether the origin and destination port of a certain demand are already *active ports* in a route, i.e., ports in which a ship already loads or offloads cargo. A port charges fixed costs per entry, so if it is active, the marginal port costs are zero. Inactive ports can be removed from a string in a later stage.

The last attribute considers the utilization of a ship between ports. Each time demand is allocated to a ship, the utilization rate of that ship between the origin and destination port of the specific demand increases. Ideally, we would like 100% utilization between all ports. This attribute considers the relative increase of the utilization rate *of the entire route* when cargo from the cargolist would be allocated. As most costs are independent of the number of containers carried, a higher utilization rate implies higher profitability. The use of *relative* increase serves other purposes as well. Routes with a lower utilization rate will be preferred over routes with a higher utilization rate. This causes a leveling effect on the utilization rates between routes. Also, routes that are already full will not immediately be filled to the maximum. The rationale is that more routes will stay candidate for assigning new demand, and local search operators will have more opportunities to move cargoes or ports, because of the higher remaining capacity.

The three attributes together are used to assign a score to each route. Obviously, the score differs for each cargo.

More formally, the individual steps of PDA Sort are as follows. For each demand we search for feasible routes (i.e., strings that contain the origin and destination port of the demand). Then we determine the value for each of the three attributes described above, where we assign the following values. For the *pathlength*, we find the shortest path between origin and destination and assign the number of ports in that path. After normalization, we take the inverse value because shorter paths should be assigned a higher score. The *activeness* is assigned value 2 if both origin and destination ports are active (i.e., existing (off)loading activities at a port), value 0 if both ports are inactive, and value 1 if only one of both ports is active. Finally, the *utilization* attribute is assigned a value between 0 and 1, equal to the potential relative increase in utilization rate by allocating the new demand to the route.

After we determined the attribute values for all routes, we normalize the three attributes using max-min normalization, to obtain values between zero and one, that easily can be compared with each other. The *pathlength* attribute values are now inversed, replacing each value by $1 - value$, because shorter paths should be assigned a higher score.

In the next step, we assign a score to each route by adding up the values of its three attributes. We then sort the list of feasible routes decreasingly by its scores, and start allocating demand to it. Similar to the Quantity Sort procedure, we allocate (part of) the demand until it has been completely allocated, or no more routes are available, each time updating the list of demands as well as the remaining capacities of the routes.

The above procedure is repeated for each demand (cargo) in the cargo list. Steps 12 to 30 of Algorithm 2 describe the procedure of the sorting methods.

4.3 Phase 2: Local search optimization

In this phase, local search optimization techniques are applied to improve each solution. Local search works by exploring neighboring solutions in the solution space, trying to discover a local optimum for each solution. To limit computational time, local search is usually subjected to a *stopping criteron*, forcing the algorithm to stop searching for better solutions when the criterion is reached. Stopping criteria are generally associated with a certain execution time, number of alternatives evaluated, or the relative improvement in the last N steps.

There are two methods to design local search operators. The first alternative is to explore all neighboring solutions, either randomly or by enumeration, constantly reevaluating the objective function. This design is straightforward and effective, but very time-consuming. The urge for stopping criteria is typically rather strong.

The second alternative is to introduce semi-intelligent operators, which search in a specific direction where you would expect the largest gain. This approach involves some sort of knowledge or reasoning to be inserted during design. For example, one operator might consider adding another port to a cycle. While an operator of the first, simple type actually adds the port and evaluates the objective functions, the second, more intelligent type, 'knows' that it makes sense to add a port only if the corresponding demand is above a certain threshold. Hence, the search for a local optimum is more directed.

We introduce three operators that try to improve initial solutions, each in their own way but all three searching in specific directions. The *route-length operator* removes ports from round trips that incur more costs than revenue, and tries to allocate unassigned cargoes by adding ports to round trips. The *port-exchange operator* relocates ports within a route or between routes in an attempt to improve solutions. Finally, the *transhipment operator* introduces the use of hubs and transhipment to save costs and allocate the remaining cargoes.

Remove inactive ports

The procedure of phase 1, where round trips are generated randomly, can cause round trips to contain inactive ports. Such ports are called, but without any (off)loading activities. In other words, a ship calls a port, pays for the port's charges, and then leaves without anything happened. That is, obviously, insensible and beyond reality. Therefore we have to remove these ports, which only incur additional costs but no revenue.

Iterating over the set of networks, we carefully analyze the routeset that is associated with each network. If there are any routes in the routeset that contain inactive ports, we remove those ports.

4.3.1 Sample service network

We introduce a small sample network that we will use throughout this section to illustrate the operation of each local search operator. The use of a small example enables the reader to better understand the rationale behind each process, while it also allows for easy comparison between individual operations.

We consider five ports, spread over three continents: Rotterdam (RO), Jebel Ali (JA), Singapore (SI), Shanghai (SH), and Tokyo (TO). These ports are represented on the map in figure 4.2. Our sample network consists of two services, A and B. The specifics of services A and B are outlined in Table 4.1. Note that these services are cyclic, after the last port in the string a ship sails to the first port of the string again. The remaining capacity between each pair of ports is based on an assumed ship capacity of 50,000 TEU per year. The cycle duration is based on the assumption that the average sailing speed is 20 knots, that ships spend by average half a day in each port for their (off)loading activities, and that the slack time is exactly one day per round trip.

In our examples, we will use the following cost and revenue parameters. Capital costs and operational costs together are assumed to be USD 2 mln per annum. Fuel costs are USD 100 per nautical mile travelled. The fixed port costs are USD 25,000, while the terminal charges USD 100 per TEU to cover handling costs. Finally, each container generates a revenue of USD 75 per day, for the sake of simplicity we do not distinguish between westbound and eastbound shipping rates.

4.3.2 Route-length operator

The route-length operator tries to add and remove ports from round trips. Both operations appear simple but require in fact a thorough methodology. In Subsection 3.2.1, we introduced different types of costs and the way in which they affect service networks. For example, removing a port from a cycle not only saves port costs, but also saves handling costs, and influences the duration of a trip and might therefore lead to the reduction of the fleet by one vessel, saving capital costs and operating costs. Obviously, similar considerations apply to the operation of adding ports to cycles.

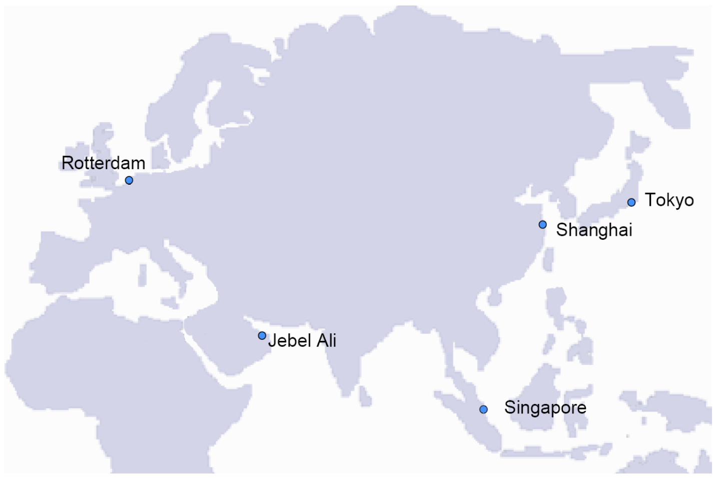


Figure 4.2: Map of sample ports

Table 4.1: Services of the sample service network (cargo in yearly TEU x 1,000)

Service A		**Service B**	
Cycle: SH - SI - RO - JA - SI		Cycle: TO - SI - JA - RO - SI - SH	
Total cycle mileage: 22,173 nm		Total cycle mileage: 23,788 nm	
Total cycle duration: 7.1 weeks		Total cycle duration: 7.7 weeks	
Allocated cargo	Remaining Capacity	Allocated cargo	Remaining Capacity
SI - RO: 25	SH - SI: 35	TO - RO: 30	TO - SI: 5
SH - RO: 15	SI - RO: 5	SH - JA: 15	SI - JA: 0
RO - SH: 10	RO - JA: 35	SI - RO: 5	JA - RO: 15
SI - JA: 5	JA - SI: 40	RO - SI: 15	RO - SI: 10
	SI - SH: 40	RO - SH: 25	SI - SH: 25
			SH - TO: 50

We start by evaluating each route to find ports that can be removed. This enumeration method considers every port of a route and calculates the total revenue that comes from allocated demand pairs which have this port either as their origin, or as their destination. Then we work out the cost side of the port. When we remove a port, we save on several cost components: port costs, handling costs for all containers that are shipped from or to that port, and fuel costs because we save some miles (the distance from A to C is shorter than the distance from A to B plus B to C, even if B lies along the route). The total costs incurred by a port are then compared to the revenue that the port generates for the carrier. If the costs are higher than the revenues, the port is removed from the cycle. Accordingly, the associated demand pairs are removed from the route and restored in the origin-destination demand matrix as unassigned cargo.

Consider Service A of our sample service network. When we remove port Jebel Ali, we save its port costs (USD 25,000 x 52 weeks), and the handling costs of 5,000 TEU (USD 0.5 mln). We also save 1.463 nautical miles (nm) by sailing directly from Rotterdam back to Singapore, instead of calling Jebel Ali on the eastbound trip. Therefore, fuel costs are saved (USD 100 x 1.463 nm x 52 weeks). Together, we save USD 9.4 mln. Now we take a look at the revenue side. The direct distance from Singapore to Jebel Ali is 3449 nm, or 7.2 sailing days. So, the generated revenue is USD 75 x 7.2 x 5,000 TEU, which totals USD 2.7 mln. Clearly, we are better off by removing Jebel Ali.

Now, in some cases, extra savings can be realized on capital and operational costs, because the fleet can be reduced by one or more vessels. This happens when the duration of a cycle is reduced such that the (rounded) duration is one or more weeks less than before. In the example above, the cycle mileage is reduced by 1.463 nm. Hence, the cycle duration is reduced by 3.5 days (3 days for the distance, and 0.5 day for time in the port). The new cycle duration is 6.6 weeks. Instead of 8 vessels, we now need only 7 vessels to maintain the weekly service, which gains us another USD 2 mln.

The second part of the route-length operator is dedicated to expanding routes, adding ports to round trips. This is possible only when there are unassigned cargoes, which can remain either from the first phase, the construction of initial solutions, or can be the result from the first step of this operator, the removal of cost-inefficient ports. In the latter case, it might seem strange to allocate these cargoes again when we previously removed them due to cost-inefficiency, but we can easily explain this. The port-removal part of this operator may have removed multiple demand pairs, with the same origin and destination port, from different routes and restored them in the origin-destination demand matrix. This means we might find profitable origin-destination port combinations again.

For example, suppose we find three routes that each carry a demand of 100 from port A to port B, and suppose we previously concluded that in all three of the cases we were better off by removing port B from the route (port A was profitable because of other demands). By removing port B, the three cargoes that were assigned to the routes were restored in the OD demand matrix. So now we have a demand of 300, to be transported from port A to port B. A demand of 300 might generate enough revenue to justify the costs involved with adding port B to a route. Hence, we have to consider all unassigned cargoes again to find profitable ones.

The process of adding ports starts with populating a list of unassigned cargoes. Next, for each cargo we walk through the set of routes and calculate how much profit we can earn if we would assign the cargo to a route (taking into account the restriction of the remaining capacities on that specific route). Some routes might already include the origin and destination of the cargo, while others may need to be extended first. If we find a port on the route twice, then we have a choice which occurrence we choose for the allocation of a demand pair. In such a case, we choose the one that will yield the smallest pathlength, so that the cargo will travel the shortest possible distance. Next, for each cargo we evaluate all routes, calculating the absolute gain in profit we could obtain when allocating a cargo. The cargo is allocated to the route that has the highest potential. The procedure of the route-length operator is summarized in Algorithm 8.

Figure 4.3 depicts an example operation of the route-length operator. It shows a sample route A with 4 ports, in which the operator both removes a port as well as adds a new port. Port 6 is removed, because the route-length operator calculated that this port incurs more costs than it generates revenue. Port 3 is added to the route, because there is enough demand from and to this port to make it profitable.

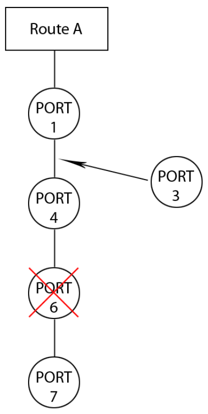


Figure 4.3: Example operations of the route-length operator

4.3.3 Port-exchange operator

The port-exchange operator tries to improve routes by moving ports. We consider both intra-route and inter-route optimization. Intra-route optimization targets individual routes, while inter-route optimization is based on a procedure that exchanges ports between pairs of routes.

Intra-route port exchange

Intra-route optimization evaluates the port sequence of individual routes. From Section 4.2 we recall that strings are generated according to a certain pattern. The ordering of a string follows the natural port sequence, but ports that appear only once in a string, apart from the two turning points which have fixed positions, can be assigned to either the eastbound part of the cycle or the westbound part. The assignment was done randomly, but now that we have more mature solutions, we can re-evaluate these assignments. Before providing the details of this procedure, we will first explain the benefits.

Let us summarize the characteristics of a cycle first. We have a sequence of ports that are called once a week, in which we load or offload cargo. Between each port we have a remaining capacity, which is defined as the ship's capacity minus the cargo we have onboard between the two ports. The cargo generates revenue, which is a fixed amount of dollars per TEU per nautical mile. To calculate the distance we use the direct distance between the origin and destination of the cargo. Costs are incurred by the distance traveled, port entrances, fleet ownership, and (off)loading activities.

Now, what happens when we move a port in the string sequence? Most parameters remain the same. In fact, all revenues and costs stay unchanged because the total distance, the number of ports, and the cycle duration do not change. Also, revenues are the same because we use the direct distance between the origin and destination of cargo, regardless of the distance the cargo really travels. The only aspect that changes is the remaining capacity between ports. There is no direct benefit from that, but in some cases the remaining capacities between other ports increase such, that new cargoes can be allocated. Hence, the most important question is, when will remaining capacities between other ports increase?

If we move a port from eastbound to westbound, or vice versa, we first have to check if this move is feasible. Often, remaining capacities in the part we are moving the port to are insufficient to allow such transitions. For example, consider Service B from our sample network. We could try to move Shanghai from the eastbound to the westbound part of the cycle. However, by doing so, the cargo from Rotterdam to Shanghai has to remain in the vessel during the port calls in Singapore and Tokyo. But in Tokyo, 30,000 TEU with destination Rotterdam have to be loaded onto the ship, so that there is insufficient capacity to carry both cargoes.

The movement of the port is only possible if all remaining capacities allow that. But even if we pass this check, we should only move the port if, in general, other remaining capacities benefit from the movement. Otherwise, there is not much use in moving the port. To check if they do, we follow a simple procedure. We compute the total pathlength of all allocated cargoes. Then we do the same, but with the port moved to its new position in the cycle. If the total pathlength decreases, we assume that, by average, the remaining capacities on the cycle increase. This provides an opportunity to allocate new cargoes.

If we look at Service B again, we might consider moving Shanghai from the eastbound to the westbound part of the cycle, because the cargo that is loaded in Shanghai with destination Jebel Ali will have a shorter travel. Previously, we showed that there is not enough remaining capacity on the other legs to move the port, but suppose the movement is actually feasible and we want to check if we gain anything from that. The pathlength from Tokyo to Rotterdam (the first allocated cargo) is 3. If we compute the pathlengths for the other cargoes as well, we get a total of 11. Similarly, we can compute the total pathlength when we move Shanghai to the westbound part, in which case we would obtain 12. So, although it appears to be beneficial to move Shanghai, the impact on the other cargoes is such that the overall improvement is negative. Hence, moving Shanghai is not only infeasible, it would also be insensible.

Now that we explained in which situations it is sensible to move a port, we can provide more details on the procedure itself. We evaluate each route successively. First, we determine which ports appear only once in a cycle apart from the turning points, which we obviously don't want to move. Then, for each single port we check the condition of the remaining capacities as describe above. Hence, we first check if the remaining capacities allow the port to be moved, and secondly, if the total pathlength benefits from the movement. If that is the case, we move the port to the opposite part of the string, so either from eastbound to westbound or vice versa. Subsequently, we update the remaining capacities. After the movement is complete, we iterate through the list of unassigned cargoes and try to allocate them to the updated route if possible. The detailed steps are provided in Algorithm 9.

Inter-route port exchange

The focus on inter-route optimization is based on the rationale that ports can sometimes be serviced by one route more efficiently than the other. Benefits can be achieved at three different levels.

First, we try to merge ports between routes. Or, more specifically, we try to allocate the demand that is handled at a certain port to fewer routes. We are particularly interested in situations where we can successfully reassign the cargo that involves a port to one route, so that we can remove that port call from another route entirely. This saves the ports costs on one route. By merging port calls, we can save port costs while the revenue remains unchanged.

Suppose we have two specific routes that each service the same port. When a route services a port, it means that the port has one or more 'counterparts', ports that are either the origin or the destination of the cargo that is handled at the specific port. Consider Service A and B again. Both services call Jebel Ali. In Service A, the counterpart port is Singapore, while in Service B it is Shanghai. But to merge both port calls, and to successfully remove Jebel Ali from one of the services, two conditions have to be satisfied. First, one of the services need to share all the counterpart ports of Jebel Ali, i.e. one of both services need to call not only Jebel Ali, but also Singapore and Shanghai. Second, the remaining capacities should allow that cargo from one service is reassigned to the other. If both conditions are satisfied we can successfully merge ports between routes. In this particular example, the first condition is not met, so that the second condition is not relevant.

The second benefit relates to the required fleet size. Recall that, because of the weekly service, we need a vessel for each week of duration. For example, if the total cycle duration is 4.5 weeks, we need 5 vessels to maintain that cycle. The key question is, how can we reduce the fleet size, and maintain the same amount of allocated cargo at the same time?

Well, suppose we have two routes with a cycle duration of 4.2 weeks each. For these two routes, we will need a fleet of 10 vessels in total. Now suppose that these two routes satisfy two conditions similar as defined in the previous paragraph, only the port we are targeting (port X) is called in only one of the route. Hence, both routes share a set of ports such that we can move port X from one route to the other, while that other route already contains the counterpart ports of port X, and the remaining capacities on the route allow for such a transition. Obviously, the duration of both routes will change. The route that 'looses' port X saves time that is otherwise spent in the port or on travelling the extra miles. Also, the slack time of a route can be reduced a little. Together, these savings can add up to several days easily. On the other hand, the duration of the route that is extended with port X will increase proportionally.

Let us assume that the first route will save 0.3 weeks (approximately two days) by removing port X. The duration of this route will then become $4.2 - 0.3 = 3.9$ weeks. Similarly, the duration of the other route becomes $4.2 + 0.3 = 4.5$ weeks. Now we require a fleet of only 9 vessels for both routes. Hence, we save capital and operational costs of one vessel. The key issue is to find pairs of routes where port movements are feasible and fleet reductions can be achieved.

Finally, the third improvement is to be found in optimizing turning points. Turning points are defined as the 'extreme' port of a cycle, i.e., the smallest and highest port number in a string of ports. Previously we mentioned that the direct distance from port A to port C is smaller than the individual distances from port A to port B, plus the distance from port B to port C, providing these ports are situated in a natural order. But the extra distance is relatively small, usually between 5-10% of the distance between A to C. For example, the distance from A to C could be 500 miles, while the total distance from A to C *via* port B could be 550 miles (with the distance from A to B being 220 miles, and from B to C being 230 miles). So, by removing port B we could save 50 miles.

However, true savings can be achieved when we leave out an extreme point, such as port C in our previous example. Naturally, we need to reassign the cargoes that are (off)loading at port C to another route. This is only sensible when port C is *not* an turning point in the other route. In such a case, we could save 230 miles in the first route, while we only have to sail an additional 50 miles or so in the other. Bottomline, we would save around 180 miles. Not only do we save fuel costs, we could also save a vessel if the duration decreases substantially. Depending on the sailing speed, sailing 180 nautical miles takes around 10 hours. Together with the time spent in a port, total cycle duration benefits approximately one day. In some cases, that might be just enough to save a vessel that would otherwise be required to service the extra port.

Obviously, we need to carefully calculate the impact on both routes and evaluate the profitability gain or loss, before we can decide to move a turning point to another route. Another condition that needs to be satisfied, is that the counterpart ports of port C also need to be present in the other candidate route.

In this subsection we specified three different levels where we can save costs, simply by redistributing ports and demand over different routes. The three techniques show lot of similarities, that is, most of the mentioned requirements are the same across the solutions. Therefore, our algorithmic approach will be more parallel. We first construct an unordered list of route-pairs. Then for each pair we evaluate the conditions of both routes and assess the appropriate operation, based on their contributions to the objection function. More specifics of the inter-route port exchange operator are outlined in Algorithm 10.

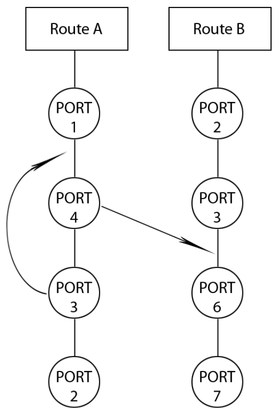


Figure 4.4: Example operations of the port-exchange operator

Figure 4.4 depicts example operations of both the intra-route as well as the inter-route optimization phases of the port-exchange operator. We see two routes, route A and route B, of which some ports are mutual and some are not. The intra-route optimization is shown on the left side, in route A. Here, port 3 is moved from an eastbound position to the westbound part of the cycle. This may happen for example because there is a lot of demand from port 4 going to port 3. In this case, the movement of port 3 will increase the free ship capacity on the remaining part of the route, which allows the allocation of other cargo.

The inter-route optimization is shown by means of the arrow between route A and route B. The arrow represents the movement of port 4, from route A to route B. Because the port is removed from route A, this route will become shorter and therefore cheaper. On the other hand, route B already calls at port 3 and port 6, and because of the natural sequencing of ports, the geographical position of route 4 is between those ports. In other words, route B already passes port 4 on its way, so that the total cycle distance will increase only a little. Such a movement is not always possible, because the 'counterpart' ports must be in both routes and there must be enough remaining capacity in the ships. However, when the movement is feasible, it can be cost-efficient.

4.3.4 Transhipment operator

The transhipment operator tries to improve the demand allocation by introducing transhipment of containers. Transhipment is defined as transferring cargo from one ship to another, with the ships operating on different services, in order to continue the journey to its final destination. The services are connected by a so-called *hub*, a port where transhipment takes place. Transhipment can take place between a feeder-line and a main-line, as well as between two main-lines. We focus on transhipment of one main-line to another main-line.

A typical transhipment move works as follows. First, ship A arrives after which the specific containers are offloaded. Then, the containers are temporarily stored at the container terminal, waiting for the arrival of ship B. When ship B is ready, the containers are loaded and proceed their journey. Naturally, every transhipment move incurs handling cost and possibly some container storage cost. In this work, we only consider handling costs, for both offloading and loading movements. As a result, one transhipment move incurs two times the handling cost per container.

Introducing transhipment can improve the allocation of demand in three ways. First, *transhipment can reduce the path length from origin to destination.* Consider our sample service network from Section 4.3.1, and suppose service B is extended on the westbound with 5 ports between Jebel Ali and Rotterdam. Now, consider the allocated demand of OD pair Tokyo-Rotterdam on service B. With the extension of service B, the path that the containers of this OD pair follow has become relatively long, i.e., making many stops. Introducing transhipment from service B to service A in Singapore yields a non-stop path between Singapore and Rotterdam. This reduces the sailing time of the containers, and most important, less remaining capacities are consumed. In the end, this will save capacity for service B at the path Jebel Ali - the 5 additional ports - Rotterdam, while only using capacity at the path Singapore - Rotterdam in service A.

Second, *transhipment can prevent unnecessary movements of cargo along the turning point.* Consider our sample service network, and suppose we want to allocate a cargo with origin Jebel Ali and destination Tokyo. Service B contains a direct path between Jebel Ali and Tokyo, but it first travels westbound to Rotterdam, before proceeding eastbound to Tokyo. Allocating to this service would yield an unnecessary move from Jebel Ali to Rotterdam and back to the Middle East, while consuming the ships capacity. Transhipment can prevent this kind of movements by connecting services. It would be much more efficient for the cargo to leave Jebel Ali in eastbound direction straightaway, as is the case in service A. Then, the cargo can be transhipped to service B in either Singapore or Shanghai, to proceed to destination Tokyo.

Third, *transhipment enables unallocated demand to be allocated.* Before transhipment is introduced, demand pairs are only allocated on direct paths, i.e., single routes that contain both the origin and destination. This means that demand pairs without a direct path are not allocated, yielding a demand matrix with remaining cargoes. This remaining demand can only be fulfilled by transhipment from one service to another. Consider our sample service network, and suppose service A is extended on the westbound with the port of Colombo (CO). The new string of service A becomes SH-SI-CO-RO-JA-SI. Now, suppose we want to satisfy demand between Tokyo and Colombo. There is no direct path between the origin Tokyo and destination Colombo. Therefore, we use transhipment to connect service B and A in Singapore. Without transhipment, this demand could not have been allocated.

An important decision that has to be made regarding transhipment is the choice which port(s) will be used as transhipment hub. Choosing the hub is a problem on its own, as we have seen in Section 2.3.5. We can choose one or multiple ports to fulfil the hub function.

With one hub, we could choose a port somewhere in the middle of the considered trade lane. When choosing two hubs, having a hub on either side of the trade lane would make most sense. For the Asia-Europe trade lane, for example, one hub could be located in Asia (e.g., Singapore), and the other one in Europe (e.g., Rotterdam). There are several approaches possible for choosing which port will be the hub, the most obvious ones are either based on the geographical location of the port, based on the current service network, or, based on the origin-destination demand matrix.

From reality perspective, the geographical location of a port determines its success as a transhipment port. A good example is Singapore that, due to its location, is the main passage to the far east, and the world's busiest transhipment port. In this approach, one could use the distance matrix to determine which port is at a central position. Perhaps the most straightforward approach is to use the current service network, and choose the port that occurs most often. The rationale behind this approach is that a high occurrence means the hub is visited by many services, which increases the number of possible transhipment moves. Another approach is to use the origin-destination demand matrix. Since the demand reflects a tangle of the port's throughput, this approach combines reality and simplicity.

However, multiple hubs seem to be the better alternative. In reality, every port can be a transhipment hub. Furthermore, the use of a single hub makes the operator less effective when the number of ports increases.

Following from the previous discussion, we distinguish three main approaches to model transhipment, corresponding to the number of ports fulfilling the hub function, and the number of transhipment moves allowed. With *single-hub single-move* transhipment, we consider only one transhipment hub, and only one transhipment move is possible. With *multi-hub single-move* transhipment, all ports are transhipment hubs, but we only allow one transhipment move. Finally, with *multi-hub multi-move* transhipment, all ports are transhipment hubs, and we allow multiple transhipment moves.

Initially, we used the single-hub single-move transhipment approach, but the improvement of the effectiveness was not satisfactory for large networks. Subsequently, we used the multi-hub multi-move approach, but this required too much computation time for large networks, i.e., the approach was not efficient. Therefore, we end up in the middle, using the multi-hub single-move approach, in order to satisfy both our effectiveness and efficiency needs. For this operator, and the experiments, we will use multi-hub single-move transhipment. A more detailed specification of the performance of the other approaches will be given as an additional experiment in Chapter 7.

The operator consists of two steps, each corresponding to a different function in the algorithm that is presented in Algorithm 11. We will discuss the precise working of each step separately.

Reroute allocated demand

The first step of the transhipment operator focuses on preventing unnecessary movements of cargo along the turning point. We only tranship demand pairs of which the current direct path unnecessary travels via the turning point. By rerouting these demand pairs to shorter transhipment paths, we prevent consuming capacities on the legs to and from the turning point. Eventually, this will yield extra space on the ships for a better allocation of demand in the second step.

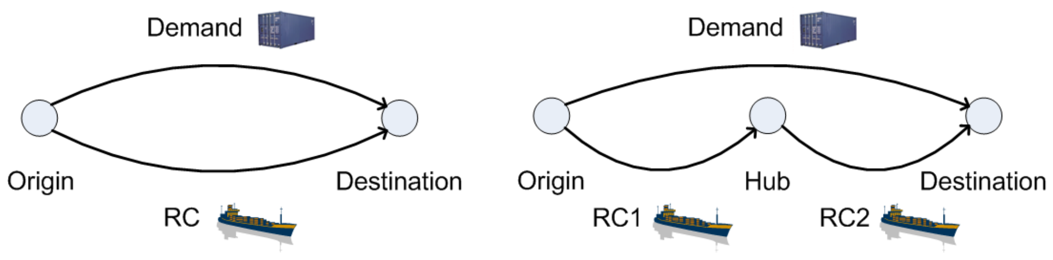
The re-routing of allocated demands works as follows. For each route in our network, we iterate trough all demand allocations. The demand allocations that are not already part of a transhipment allocation, and currently have a path that unnecessarily visits a turning port are considered for further investigation. For these demand pairs, we search candidate transhipment paths according to the the multi-hub single-move transhipment characteristics. This means we consider all ports within the interval [origin, destination] to be the hub port, one at a time.

For each of these possible hubs, we iterate through the routes that contain the hub, searching for the origin and destination port. We search for both origins and destinations, to obtain origin-hub paths as well as hub-destination paths. After we have iterated through the routes we merge the origin-hub and hub-destination paths, obtaining all possible transhipment paths for a specific hub port. Subsequently, we continue our search for the next hub port.

At this point we have one list with all possible transhipment path candidates, via different hub ports. We sort this list according to an ascending path length, i.e., the number of stops of the considered candidate. Then, we can start our reallocation process. Recall that the list of candidate paths corresponds to a single demand pair we wanted to reroute. We first try to reallocate the demand to the shortest path. However, the maximum demand we can re-route to this path is equal to the smallest of the remaining capacities (RC) of the origin-hub part and hub-destination part of the candidate. This is visualized in Figure 4.5, where the maximum demand to reroute is min(RC1,RC2). We try to re-route as many demand as the remaining capacities of the candidate allow. When we fail to allocate the full demand to the first candidate, we continue with the next candidate, and repeat this process until we have run out of demand or candidates. The pseudo code for this step of the transhipment operator is given in Algorithm 12.

Allocate demand matrix leftovers

The second step of the transhipment operator tries to allocate demand pairs from the demand matrix, that we could not satisfy with direct paths. The introduction of transhipment allows these pairs to be allocated as well. Just like in the previous step, the amount of demand we can allocate depends on the remaining capacities of the transhipment paths. This step is processed after we rerouted existing demand pairs, in order to have enough space to reroute. When we would first allocate the demand matrix left-



(a) Normal situation (b) Situation with transhipment

Figure 4.5: Direct path versus transhipment path

overs, we would consume remaining capacities, making it more difficult to reroute the already allocated demand pairs. Moreover, the rerouting of existing demand yields more transhipment possibilities for the allocation of demand matrix leftovers.

The allocation of demand matrix leftovers works as follows. It is analogous to the rerouting of allocated demand, except now we iterate through the demand matrix instead of the allocated demands of each route. For each row of the demand matrix, we iterate trough all columns. Whenever there is demand left for the considered origin-destination pair, we search for candidate transhipment paths. Again, we obtain a list with all possible transhipment path candidates, via different hub ports that lay within the interval [origin, destination]. We sort the list by ascending path length, and start allocating the demand to the first candidate, and proceeding similarly as the previous step of the operator. The pseudo code for this step of the transhipment operator is given in Algorithm 13.

4.4 Conclusion

In this chapter, we presented the multi-start local search algorithm for the routing and scheduling problem in liner shipping. We are now able to answer the third research subquestion:

3. *Can we transform the multi-start local search algorithm from tramp shipping studies to the case of liner shipping?*

Multi-start local search is a heuristic that combines two techniques, i.e., the multi-start technique and a local search heuristic. These two techniques represent separate phases in the multi-start local search algorithm. The first phase can be seen as an insertion heuristic that generates multiple initial solutions. The second phase tries to improve these solutions using a local search heuristic. The advantage of a multi-start approach is that the solutions are spread throughout the solution space, thereby decreasing the chance of getting stuck in a local optimum.

The multi-start local search approach is a general technique, and not restricted to a specific problem. The implementation of the two phases determines the applicability of the algorithm for a specific problem. In our case, the starting point was research that used multi-start local search for solving the routing and scheduling problem in tramp shipping. Recall that liner shipping companies provide services according to a fixed network of routes and a regular repetitive schedule, whereas tramp ships have no fixed network and schedule, and mainly trade on the spot market. As a result, the assumptions underlying the problems differ, yielding a different implementation.

We were able to transform the multi-start local search algorithm from tramp shipping to liner shipping by creating such a different implementation. Since liner ships operate on a route that is fixed for a long time, it is essential to find proper routes that can satisfy future demand. Whenever a network comes into place that does not yield a reasonable utilization, the carrier loses money. Therefore, the process of deciding on the network to service needs to be very accurate. The most important difference between the tramp shipping and the liner shipping implementation is that tramp shipping moves with spot cargoes to find the optimal routes, whereas liner shipping mainly moves with ports to find the optimal network. This is also expressed in the route-length and port-exchange operator, that move with ports to find a better network. The transhipment operator does not change the network but rather the allocation of demand. Still, the operator is not applicable to tramp shipping since transhipment is not an issue when there is no fixed network of routes. Furthermore, the sort methods used in the first phase of the heuristic differ as a result of the approach. The *time sort* method used in the tramp shipping implementation is only applicable to tramp shipping, instead we introduced the *profit-driven sort*. The *quantity sort* method had to be changed to cover for the difference between single spot cargoes from tramp shipping and the aggregated yearly demand between ports from liner shipping.

In summary, we can state that we have transformed the algorithm from tramp shipping to the case of liner shipping, by providing a different implementation, that follows from the different assumptions that liner shipping is based on.

Chapter 5

The Maersk Asia-Europe data set

This chapter deals with finding a data set that combines reality and general applicability. The research subquestion we try to answer is:

4. *What is an adequate data set to assess the effectiveness of solutions to the problem?*

Section 5.1 gives a general introduction to the characteristics of data sets for the routing and scheduling problem in liner shipping. Furthermore, we discuss existing data from literature and elaborate on their usability. Section 5.2 presents a new data set and gives an explanation on how it was constructed.

5.1 Introduction

Benchmark studies add more value when standard data sets are used, that allow for good comparison between the performance of different solution approaches. However, due to the numerous variants of the problem, there is no de facto standard data set available in the field. This leaves researchers with no other choice than to collect or generate some data themselves, and create their own case study. In this section we will discuss some existing secondary data sets and their fit to our problem, but first we focus on the characteristics of a suitable data set.

The objective for a standard data set would be to combine general applicability and reality. General applicability means that the data set is usable in different variants of the problem. This can be addressed in two ways, either by providing a broad data set covering all variants (i.e., a data set that has all possible variables, also with data not necessary for some variants), or by providing a compact data set with basic variables that are needed in any problem variant. In the latter case, researchers have to generate additional data to fit their variant of the problem. From the description of the routing and scheduling problem it is clear what variables are necessary to incorporate in a data set. The most important variables are the demand, fleet, the distances between ports, revenue, and costs. These variables need to be specified for any variant of the problem.

A problem we face when pursuing general applicability is the format of the variables. For example, we have seen data sets with a list of origin-destination demand pairs, whereas others use origin-destination matrices. Naturally, this does not influence the contents of the data and is therefore considered to be unimportant. However, the unit of measurement is important. As an example, it is common to measure distance between ports in nautical miles, but some studies do not consider distances in length, but rather use time units to indicate the time needed to make the travel. Furthermore, demands can be measured in TEU or FEU (i.e., forty-foot equivalent units), or even in tons.

Apart from general applicability, a data set should incorporate reality. Realistic data sets provide more appealing examples, and allow the results to be placed into perspective with the real world. It is very difficult to obtain real-world data, since carriers treat their operating data as confidential. Nevertheless, carriers do provide some data through their annual reports and their websites, that contain useful information about the operations of the company. General sector data (e.g., port throughput, price levels) are available from several online resources ranging from specific websites to magazine databases.

Still there exist some problems with respect to obtaining the data. For example, there is no information available on the exact demand between ports, but rather on the major trade lanes Europe-Asia (Song & Carter 2009), Trans-Pacific (Song & Carter 2009, Fusillo 2004), and Trans-Atlantic (Song & Carter 2009, Fusillo 2004). Carriers also provide only the total demand on trade lane level. Even when the demand between ports would be known, the question remains whether the data represent the underlying values, since supply creates demand. Furthermore, demand between ports is measured per year, while in reality seasonal demands occur. Another danger concerns the level of revenues and costs. Since these variables are constantly fluctuating, old data (e.g., from related work) might not reflect todays reality.

In liner shipping literature, a few related studies have used data to present results of their solution approach. Some used real-world data, others generated random data, and some used both.

Alvarez (2009) presents a case study using realistic data, albeit not related to any carrier in particular. The top 120 worldwide ports by throughput were chosen to use in the data set. Distances were computed based on data from the U.S. National Imagery and Mapping Agency. However, accuracy tests using several online distance calculators show substantial differences in the reported distances, with the conclusion that the presented distances are unreliable. Based on an average total annual demand of 5 million TEU, countries of considered ports were assigned demand based on the countries gross domestic product (GDP). One can argue whether the GDP is a good criterion to distribute the total demand, since it leads to totally different container streams as seen in reality. Next, based on ports throughput ranking, the country demand was assigned over the ports. Then, port demand was distributed over other ports using a logit model, based on hinterland demand and sailing distance, to obtain origin-destination pairs. To model the fleet, 100 vessels were generated based on 5 different vessel classes, based on their size (1000-8400 TEU). The operating costs of these vessels were related to their specific class, i.e., different speeds for each class leads to different fuel costs.

Agarwal & Ergun (2008) consider two regions and randomly assign ports to either region. The distances were randomly generated based on average sailing times on Trans-Pacific routes. Demand pairs were randomly drawn from the set of ports, and assigned a demand between 0.1 and 1.0 times the capacity of the largest vessel. The fixed fleet is based on 3 vessel sizes (2000, 4000, 8000 TEU), and the vessels are randomly chosen. Revenues are considered proportional to the distance between ports, but a proportionality constant is randomly drawn from [100,200] to prevent always taking the longest routes. Here, the authors do not consider capital costs, since these are fixed during the planning period. Operational costs used are port costs, proportional to the vessel size, holding costs for storing containers, and genuine operation costs (e.g., fuel costs), depending on the vessel size and distance travelled. The rather large random aspect of this data set means that it does not represent reality, and is therefore not an optimal case.

Yan et al. (2009) obtained data from an unspecified major Taiwanese liner shipping company. This study focuses purely on scheduling, and therefore the case example already consists of two services, based on real intra-Asia services. Their goal is to find a time schedule that minimizes costs. The demand between ports was known from the company. The fixed fleet consists of vessels of 2 sizes (870 and 2500 TEU). Furthermore, actual cost parameters of both ship sizes were used, for example the operating costs were \$833.75/day and \$2,395.83/day. Port costs range from \$5,000 to \$15,000. The container handling costs at the ports range from \$80 to \$220 per TEU. As in the previous example, container holding costs are considered, ranging from \$2 to \$5.5 per TEU per day. The data used in this example give some reasonable cost ranges, although the prices fluctuate and differ among ports.

5.2 Data set description

In the previous section we have seen the difficulties with the current data that is used in routing and scheduling for liner shipping. Our goal is to find a data set that combines reality and general applicability. Next, we present a new data set that strives to meet those requirements.

In order to address the reality constraint, we use real data from various sources, and base our data set on an existing service network. The main rationale for choosing an existing service network is that it provides for a highly-relevant comparison. That is, comparing the best service network we obtain using our algorithms, with the service network as it is actually implemented by the carrier. Assuming the carrier did its utmost best to compose its service network, the comparison places our solution into perspective, and when the results are good, justifies the use of our algorithms. We have chosen the service network of Maersk on the Asia-Europe trade lane. This is one of the three major trade lanes, besides Trans-Pacific and Trans-Atlantic. Focusing on one trade lane seems the best option, since many data is decomposed to this level, and it keeps the data set from growing too big, as would be the case when considering worldwide services.

To address the general applicability, it is necessary to consider several problem variables. Based on the small literature review with respect to data usage, we choose the variables that are necessary to make the data set applicable to most of the related work. Naturally, this includes an origin-destination demand matrix between considered ports. Often, this is used together with a matrix of distances between ports. Another common variable is the fleet of vessels available to perform the services. Furthermore, revenues and costs should be specified, corresponding to the actual values seen on the Asia-Europe trade lane.

5.2.1 Original service network

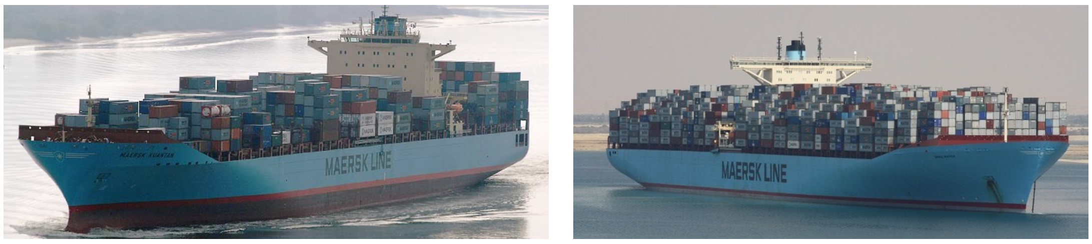
The data set we present is based on the original service network of Maersk for the Asia-Europe trade lane during spring 2010. At the time, its service network consisted of the services AE1, AE2, AE3, AE6, AE7, AE9, AE10, AE11, and AE12. Originally, additional services AE21 and AE23 existed, which were slot chartering services on CMA CGM France-Asia Line (FAL). However, these services were terminated and replaced by a new joint service with CMA CGM named AE8 in the summer. Other joint services with CMA CGM are AE3, AE11, and AE12. In the appendix, Tables B.1 and B.2 present the string of ports for the beforementioned services, as well as the vessels that operate on these services. This data was publicly available from the Maersk Line web site. In the next subsections we discuss the specific problem variables included in the new data set.

5.2.2 Ports

We aggregated all services to obtain one list of ports, shown in Table B.3. Note that we removed Los Angeles (AE6) from the list, since it is not on the Asia-Europe trade lane. In total, this amounts to 58 ports. Using the Container Traffic World Ranking 2009 of Containerisation International, and port web sites, we looked up port specific information like it's code, country, continent and container throughput.

5.2.3 Fleet

We obtained the fleet by aggregating the vessels of all services. We looked up their capacity using a list with the world's fleet from Containerisation International. Since we removed Los Angeles from the list of ports, we also removed 2 vessels (Maren Maersk & Marchen Maersk) to compensate for the decreased sailing time of the AE6 cycle. The fleet also consists of 11 CMA CGM vessels that operate in joint services AE3, AE11, and AE12. The total fleet amounts to 91 vessels, and is shown in Appendix Table B.4. The vessel size ranges from 6,251 to 14,770 TEU. Figure 5.1 illustrates this difference. Naturally, the information in this variable only applies to problem variants that consider a restricted fleet.



(a) Maersk Kuantan (b) Emma Maersk

Figure 5.1: Difference in vessel size: (a) 6,500 TEU versus (b) 14,770 TEU

5.2.4 Demand

The demand between ports is determined in the following way. First, we determine the total demand to be allocated on the Asia-Europe trade lane. According to the annual report 2009 of A.P. Moller - Maersk Group, the total container demand fulfilled by Maersk Line was 13.8 million TEU. The 2010 interim report indicates a 11% growth in volume for the first half of 2010 with respect to the same period in 2009. Based on this information, we also expect the total yearly demand to be 11% higher, yielding 15.318 million TEU. From the 2009 annual report we know that the share of the Asia-Europe trade lane is 40%, which we expect to remain the same, yielding a total demand of 6.13 million TEU.

Additionally, we want to correct for the CMA CGM vessels in joint services AE3, AE11, and AE12. On these services, CMA CGM and Maersk both are responsible for a part of the total capacity. Since we consider the latter services as part of the Maersk service network, and the demand handled by CMA CGM is not included in the previous estimations, we have to add more volume. Since the distribution and capacity of Maersk and CMA vessels on the services is not equal, the amount to be added is based on the share of the total demand for one TEU capacity unit.

We first divide the total demand (6.13m TEU) by the total capacity of the Maersk ships (695,126 TEU[1]) to obtain the share of demand volume for 1 TEU capacity unit. This yields 8.8145 TEU per TEU capacity unit per year, i.e., on average, 1 TEU capacity unit on a vessel (1 slot) is used by 8.8145 different containers in one year. Then we determine the share of the CMA vessels on these services by multiplying their capacity (73,433 TEU[1]) with the demand volume per TEU capacity unit. As a result, the considered CMA CGM vessels are expected to handle 647,280 TEU and the final total yearly demand becomes 6.78 million TEU.

Now that we have determined the total demand to allocate on the trade lane, we can start distributing the demand over port-port combinations. The port-port combinations are all possible origin-destination pairs based on the list of ports. Since we want the distribution to reflect reality, we cannot just randomly assign demand to origin-destination pairs. Instead, we base our distribution on the throughput of the ports. Since an origin-destination pair consists of two ports, and the demand assigned to pair A-B differs from the amount assigned to B-A, we distribute the demand in two steps.

First, we distribute the total trade lane demand over the 58 ports from the list of

[1]based on Table B.4

ports, based on the port's throughput. Now, each port has a different fraction of the total demand assigned to it. One can think of this 'port demand' as one-sided demand, only representing the origin in an origin-destination pair.

Second, for each port, we distribute its 'port demand' over candidate destinations to obtain full origin-destination pairs. These candidate destinations have to be ports that satisfy one constraint, the port should lay in another region than the origin port. Note that this constraint ensures we only serve interregional demand. Technically speaking, this step comes down to creating a subset of destination ports that satisfies the constraint, after which the 'port demand' is distributed over the destination candidates in the subset based on the destination's port throughput. The result is one of the 58 rows of the demand matrix. This second step is performed for all 58 ports, to obtain the full demand matrix.

Whether the port's throughput yields a correct prediction of the origin-destination demand can be point of discussion. For example, the throughput of a port also includes empty container handling. However, as long as the percentage of empty containers as part of the total throughput is the same at all ports, this is no problem. Another factor included in port throughput is transhipment moves. One port can have a larger percentage of transhipment moves than another, thereby falsely assigning more demand to the port. Demand between port A and C, that is transhipped in port B, is now also assigned to port B. As a result, in our distribution of demand, port B would have a greater share than it deserves. In this way, transhipment moves are integrated in our demand data. What originally was demand between A-C, will become demand between A-B and B-C. Nevertheless, we believe the port throughput is the best approximation of reality among the statistics available.

Page 190 in Appendix B shows the demand between ports for our data set. Aggregating the demand of OD pairs with westbound direction yields 4.88m TEU, equal to 71.97%, and the eastbound direction is responsible for 1.90m TEU, equal to 28.03%. This indicates a substantial trade imbalance on the Asia-Europe trade lane.

The demand matrix, as well as the distance matrix, is displayed in alphabetical order. However, we assume that the ports lay in a natural order, such that the complex traveling salesman problem is avoided. In the digital matrices, the order is shifted to fit the natural order. The natural order of the list of ports is presented in Table B.7.

Algorithm 1: Obtaining distances from PortWorld Distance Calculcator

```
Input: MySQL database thesis with table reqports, consisting of one column portname,
       containing the required ports.
Output: Table distances with three columns, from, to, and distance.
<?php;
// Set up the MySQL database connection
mysql_connect("localhost", "root", "");
mysql_select_db(thesis);
// Create array of ports
var $ports = array();
var $result = mysql_query("SELECT portname FROM reqports ORDER BY portname ASC");
while $row = mysql_fetch_object($result) do
    array_push($ports, $row → portname);
end
// Generate array with all port-port combinations
var $comb = array();
for $i \leftarrow 0$ to count($arr)-1 do
    for $j \leftarrow i + 1$ to count($arr) do
        $comb[count($comb)] = array($arr[i], $arr[j]);
    end
end
foreach $comb as $c do
    // Generate url to fetch
    var $a = "?fromport=" . $c[0] ."&port=" . $c[1] .
    "&lat=NaN&lng=NaN&bos=true&suez=true&pana=true";
    $a = str_replace(array(" ", "(", ")"), array("%20", "%28", "%29"), $a);
    var $url = "http://www.portworld.com/map/map-route.php" . $a;
    // Set up PHP Curl environment for fetching websites
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL,$url) ;                        // set url to post to
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1) ;                // allow redirects
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1) ;                // return in variable
    curl_setopt($ch, CURLOPT_HEADER, 0) ;                      // return no http header
    // Fetch url and store distance in database
    $result = curl_exec($ch) ;                                      // execute fetch
    curl_close($ch);
    $b = json_decode($result) ;                             // convert to PHP variable
    echo $c[0] . " ⟶ " . $c[1] . ": " . $b → totalmiles . " nautical miles<br/ >";
    $result = mysql_query("INSERT INTO distances('from','to','distance') VALUES (' " . $c[0]
    . "', ' " . $c[1] . "', ' " . $b →totalmiles . "')") or die(mysql_error());
    flush();
end
?>
```

5.2.5 Distances

Finding the distance between ports turned out to be a major challenge. On the internet, several web sites have distance calculators to compute single distances. However, with 58 ports, we need to retrieve 1653 unique port combinations. Provided that manually performing this task will take some time and probably cause errors, we decided to come up with an automated solution. After some investigation, we found a distance calculator website suitable for such a solution, PortWorld Distance Calculcator. Using a self-constructed PHP script, and our list of ports, we managed to obtain 90% of the combinations.

The 10% missing values were caused by missing ports and database errors. Some ports were included in the website's database, but did not contain any distance values. These values had to be added manually using different web sites, Distances.com Distance Calculcator and Sea Rates Distance Calculcator. Among these web sites, different values for the same pair can be found, due to different methods of distance calculation. However, the differences are not substantial. For example, for the distance Rotterdam - Hong Kong, PortWorld Distance Calculcator found 9,668 nm, Distances.com Distance Calculcator 9,742 nm, and Sea Rates Distance Calculcator 9,684 nm.

The PHP script is presented in pseudo code in Algorithm 1. The result of the PHP script is a list of port-to-port distances. We assume the westbound and eastbound directions do not differ in distance. The final distance matrix is presented on page 191 in Appendix B.

5.2.6 Revenues & Costs

A specification of the revenues acquired with transporting containers and the costs incurred during this activity are part of our data set. However, since the revenue and cost variables are easier to adjust than the fleet, demand, and distance variables, we specify them with the additional model variables in Section 6.2.1.

5.3 Conclusion

In this chapter, we discussed the characteristics of a data set for the routing and scheduling problem in liner shipping. The required characteristics for a suitable data set are reality and general applicability. In this regard, general applicability refers to the broad usability of the data set in different problem variants. We reviewed data sets from related work and presented a new data set. The research subquestion we tried to answer is the following:

4. *What is an adequate data set to assess the effectiveness of solutions to the problem?*

The three example data sets that we discussed did not meet the requirements of reality and general applicability. They were either not based on reality, or did not contain enough data to be general applicable. The data set from Alvarez (2009) contained unreliable distances and used the GDP to distribute total demand over the ports. Agarwal & Ergun (2008) used random values for several model variables, thereby not fulfilling the reality aspect. Yan et al. (2009) used private carrier data and only provided some information on cost levels.

With the proposed Maersk Asia-Europe data set we strive to meet the requirements of a suitable data set. It is based on the Maersk Asia-Europe service network during spring 2010, and contains 58 ports. The data set consists of a demand matrix, distance matrix, fleet, and revenue and cost information. We used several resources to generate a representative and complete data set. We believe the MAE data set is adequate to assess the effectiveness of solutions to the routing and scheduling problem in liner shipping.

Chapter 6

Experimental design & results

This chapter outlines the design of our benchmark study and reports on the results of the experiments. In particular, we provide answers to the following research subquestions:

5. *Does the more advanced profit-driven sort insertion heuristic contribute more to the quality of the solutions in the multi-start heuristic than the simple quantity sort insertion heuristic?*

6. *What local search operators contribute the most to the objective function?*

7. *What is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping?*

8. *Under what circumstances is the use of transhipment hubs effective?*

Section 6.1 gives a general introduction to the benchmark study we conducted. Section 6.2 outlines the design and experimental setup, while Section 6.3 presents the results of the experiments. Finally, Section 6.4 provides an analysis and interpretation of the results.

6.1 Introduction

In Chapter 4 we introduced the two-phased multi-start local search algorithm for the routing and scheduling problem in liner shipping. In the first phase, the *insertion heuristic*, we generate random solution candidates. The majority of the demand is allocated to those candidates using two sorting methods: Quantity Sort, and PDA Sort. We can refer to the Quantity Sort method as a basic, straightforward approach. PDA Sort, however, tries to incorporate a more intelligent view on the demands.

The second phase is the *local search heuristic*. In this phase we try to improve solution candidates by exploring the neighborhood of each candidate in an attempt to find its local optimum. Here, we use three different local search operators: the route-length operator, the port-exchange operator, and the transhipment operator which introduces transhipment to the solutions.

We can combine each local search operator with another, as well as with different sorting methods. Each combination is valid, as long as we use at least one sorting method in phase 1, and one local search operator in phase 2. All together, this provides 21 different combinations. When we examine different sequences of the local search operators (i.e., does it matter in which sequence we apply the local search operators?) the number of possible combinations is even larger. When we compare the results of different combinations, this gives us insight into the performance of individual operators. It will also reveal the best configuration to solve routing and scheduling problems in liner shipping, using the multi-start local search algorithm.

We introduce a benchmark study that runs and compares all possible combinations of sorting methods and local search operators. In section 6.2, we outline the design of this study.

6.2 Experimental Design

The use of benchmark studies to compare algorithms with respect to a certain performance measure is an established exercise. A proper framework to conduct such studies, and different methods to apply statistical test procedures have been introduced by Hothorn et al. (2005). Benchmark studies are particularly suitable to assess the performance of different learning algorithms, such as regression or classification models. For these type of algorithms one can use resampling methods, such as cross-validation and bootstrapping, to compare estimates of generalization error among different algorithms.

Although we cannot use resampling due to the nature of our data, we use the basic framework to conduct a benchmark study. This means we run different implementations of our algorithm using a single data set, and compare the values of the output variable, the network profitability.

First, we present the model variables and their settings for this particular benchmark. Subsequently, we do the same for the algorithmic parameters. After that, we shortly discuss the data set that is used and elaborate on the implementation of the benchmark study.

6.2.1 Model variables

The multi-start local search algorithm is used to solve the routing and scheduling problem in liner shipping. We need to specify the variables that are related to the economical problem, such as sailing speed and cost parameters. Although a qualitative assessment of the algorithm does not depend on the exact setting of these variables, the outcome will obviously be different when the settings are altered. We provide the variables and their settings used in this experiment, so that the benchmark study will be verifiable and reproducible. The settings are chosen to represent reality, even though the details of some variables, such as revenue, are often difficult to obtain, and have to be estimated based on the information that is available.

Sailing speed

The sailing speed is defined in knots (nautical miles per hour) and is considered to be constant in this study. In reality, sailing speed is varied mainly to control fuel consumption. In this area, a fairly recent trend is known as *slow steaming*, a reduction of the sailing speed to compensate for fleet overcapacity. Although more realistic, the use of speed variation actually provides little more insight into the problem and its solution, while it complicates the algorithmic design. Hence, we use a fixed sailing speed of 20 knots, which seems fair since modern vessels are able to sail up to 30 knots.

Ship capacity

A fleet is usually composed out of different types of ships, each with their own cargo capacity. Larger ships typically have a lower cost per TEU per mile, although their capital costs are substantially higher. Therefore, the use of large ships must be justified by a higher demand. In other words, utilization rate must be healthy to ensure the use of large ships is economically viable. In algorithms, the use of different ship capacities is complex, or computationally intensive at the least. One has to solve the problem for different combinations of ship capacities, each time evaluating the objective function.

To avoid this problem, we use a fixed ship size. We assume a homogeneous, unrestricted fleet of vessels, with a capacity of 12,000 TEU. We choose this rather large capacity because it seems to fit the recent trend for ultra-large container ships. Also, we consider inter-regional demand only, which (at least on the Asia-Europe trade lane) involves large demands.

Capital and operational costs

Owning and operating a fleet incurs capital and operational costs. Capital costs depend on the size and capacity of the ship. Operational costs are related to a ship's size as well, because they typically involve maintenance, insurance and wages. Therefore, operational costs are sometimes expressed as a fraction of the capital costs.

The relationship between a ship's size and its capital costs is non-linear. In Van der Meer (2011), a function is introduced that provides a relation between a ship's size c (in TEU) and the purchase price P (in US dollars):

$$P = 100,000c^{0.75} \tag{6.1}$$

This relation is based on historical purchasing data, acquired from internet sources (prices as of 2010). When we use this function to estimate the purchasing costs of a 12,000 TEU vessel, we obtain the amount of \$ 115m.

Based on expert estimates at the Port of Rotterdam, Van der Meer (2011) also suggests that average fleet lifecycles are 10-15 years. We use the mean (12.5 years) to compute the yearly costs associated with the purchasing price. To account for the expected return on future cashflows, we have to apply a discount rate using equation 6.2.

$$\begin{aligned} \bar{C} = \frac{P}{\sum_{t=0}^{T} \frac{1}{(1+r)^t}} &= \frac{r \cdot P}{(1+r) - (\frac{1}{1+r})^T} \\ \text{where } \bar{C} & \text{ denotes the average costs per year} \\ T & \text{ denotes the expected average lifecycle of a ship} \\ P & \text{ denotes the purchasing price of a ship} \\ r & \text{ denotes the expected interest rate} \end{aligned} \tag{6.2}$$

We assume that the average interest rate is 5%. Using $P = \$115m$, $r = 0.05$, and $T = 12.5$, we obtain the average costs per year, $\bar{C} = 11,350,574$. Hence, we assume that the average capital costs per year equal \$ 11.35m.

Cullinane et al. (1999) showed that operational costs can roughly be divided into insurance costs on the one hand, and wages and other operational costs on the other hand. Drewry Shipping Consultants (2007) uses a more refined cost breakdown into crew, insurance, repairs and maintenance, stores and supplies, and management and administrative costs. The latter report provides projected operating costs for different types of ships. In bulk carrier ships, the type that we mainly consider in liner shipping, one often distinguishes between *Panamax* and *Cape-size* ships, referring to the maximum size of a ship for it to be able to pass the Panama canal. A ship that is too large to use the Panama canal must sail around Cape of Good Hope or Cape Horn, hence the name. The type of container ships we consider are *Suezmax* ships, these are ships that have an acceptable draught for passing the Suez Canal. These kind of ships usually have a crew of around 14 persons on board. For this type, Drewry Shipping Consultants (2007) projects the average operating costs to reach US$ 6,000 per day in 2010. We assume that a ship generally is operated 365 days a year, so that the operating costs are $ 2.19m.

Fuel costs

Fuel costs (also known as 'bunker costs') are very important because they represent a large portion of shipping companies' budgets. Because of heavily fluctuating fuel prices, liner conferences introduced a so-called 'bunker adjustment factor' (BAF) formula. A shipping company typically charges basic shipping rates, with surplus charges to account for fuel price fluctuations. While the basic shipping rate is relatively steady and is mostly influenced by market conditions, the BAF uses monthly updated fuel price indicators. In this way, ocean freight pricing is more transparent and shipping liners are able to recover a larger portion of bunker costs from their customers. As liner conferences related to European trade expired in 2008, Maersk Line introduced its own BAF formula (Maersk Line BAF 2010).

To model fuel costs in this study, we simply use a fixed sum of $150 per nautical mile. Although this figure varies not only with the fuel price, but also with the type of vessel and the sailing speed, it appears to represent most related studies, such as the ones from Shintani et al. (2007) and Notteboom & Vernimmen (2009).

Port and handling costs

Port costs usually vary among ports, mainly because of competitive reasons. Port costs also vary with the type of vessel, where ship capacity, length and draught are the main cost drivers. Because the differences are relatively small compared to the total costs a shipping company faces, we assume that port costs are fixed. Data from different studies indicate that average port costs vary between $ 15,000 and $ 30,000 (Shintani et al. 2007, Ting & Tzeng 2003). We use a fixed sum of \$ 25,000 per entry, regardless of the port and the type of vessel.

Handling costs are charged by the terminal for (off)loading activities. While each terminal charges its own fee per container, average fees appear to be in the \$150 - \$200 range (Shintani et al. 2007, Ting & Tzeng 2003). Therefore, we use a fixed sum of \$175 per TEU.

Revenue

The revenues are based on historical freight rate data from Containerisation International, shown in Figure 6.1. The graph indicates the difference in westbound and eastbound rates, where the westbound (Asia-Europe) rates are much higher, and shows the rates are quite asymmetric. Note that these freight rates are average rates, based on an average trip along the Asia-Europe trade lane.

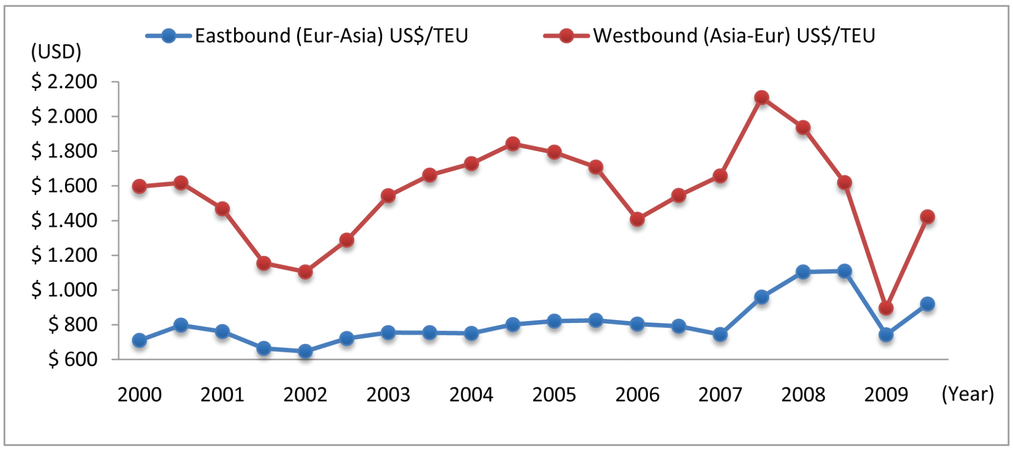


Figure 6.1: Historical freight rates on the Asia-Europe trade lane

We choose the westbound and eastbound rate to correspond to the 10-year average, i.e., \$1,400/TEU westbound, and \$700/TEU eastbound. Of course, these rates still have to be divided by the average distance on the trade lane to obtain the rate per nautical mile. From our data set, we have calculated the average distance between Asian and European ports, which turns out to be 8,350 nm. This leads to the conclusion that the average revenue per nautical mile is \$0.1677/nm westbound, and \$0.0838/nm eastbound. We use these rates in our model to represent the revenue.

Slack time and port delay

Slack time is defined as built-in, extra time to cover for delays on the route, due to for example weather conditions or delays at ports. Ships may arrive in a port in a certain time slot, but must often pay a 'penalty' when it arrives too late. To avoid this, and to ensure that the weekly arrivals at each port are maintained, slack time is used. We use one day slack time for each route, independent of a route's length or duration.

The time a ship spends in a port actually depends on more factors, like the time of arrival, working-load of the terminal, the number of containers that have to be (off)loaded, and the number of cranes used for the (off)loading activities. Because these facts are uncertain and difficult to model, we use a fixed time of 0.5 days per port call.

In Table 6.1 we have summarized all model variables.

6.2.2 Algorithmic parameters

Besides the variables used in the model, there are algorithmic parameters as well. These parameters do not relate to the economical problem but are used to control the algorithmic process. Below, we define the parameters and their settings. Some parameters are defined as *interval*, which means that we let the algorithm choose random values within a certain interval. We do this because it ensures diversity, while we avoid insensible values (e.g. we avoid cycles with only one port).

Number of service networks

The multi-start local search algorithm works by repeatedly generating service networks and improving them locally. The number of service networks that have to be generated is manually set. For each run in our benchmark study, we build 100 service networks.

Number of routes in each network (interval)

Each network is initialized by randomly creating routes. To ensure variability among the different networks, the number of routes that is created in each network should be different. Creating a random number of routes each time will certainly ensure diversity, but we have to draw a random number from a limited set, otherwise we could face either extremely small or extremely large random numbers.

The interval we choose is rather important because it will affect the performance of the algorithm, both in terms of efficiency and effectiveness. If the upper bound is too high, the algorithm will probably be slow because it has to work through a large number of routes. However, when the lower bound is chosen too low with respect to the number of ports in the data set, each network is likely to contain too few routes to ensure diversity among them.

Diversity in this matter is essential because, ideally, we want each port to appear regularly in the set of routes. The possibilities of adding ports to a route in a later stage is rather limited, since the remaining capacities on a route often prevent unlimited expansion. It is considerably easier to remove unprofitable ports from a route.

All in all, the safest approach is to choose a wide interval, so that networks with both small and large numbers of routes will be generated. The procedure of the algorithm is such that, obviously, only the strongest network will survive.

However, because we have the knowledge mentioned above, we choose to fit the interval to this specific data set. The algorithm will draw a random number of routes from the uniform distribution [6,15].

Length of a cycle's string (interval)

Ideally the length of a string should be unbounded, that is, only the theoretical limits should be applied. In other words, a string can have a length anywhere between 2 and $(2 * p) - 2$, where p is the number of ports in the data set. However, for larger data sets, one can choose to enforce a decreased upper bound, for example just p. This serves two purposes. On the one hand, it limits computational time because the strings in general will become shorter. On the other hand, real life examples show that string lengths are usually rather modest compared to the number of ports in a data set. Although one does not want to constrain an algorithm too much, modest boundaries won't be harmful.

Because we use a rather large data set, we choose to enforce boundaries that are tighter than just the theoretical limits. The interval we use is between 2 and 30, whereas the theoretical upper bound would otherwise be 114. The final length of a string is determined by drawing a random number from the uniform distribution [2,30].

Fraction to randomly allocate in phase 1 (interval)

We have outlined the different phases of the multi-start local search algorithm in Chapter 4. In phase 1 of the algorithm, for each network, we generate several routes, and allocate part of the demand to it in order to ensure diversity. After that, the remainder of the demand is allocated using the two sorting techniques.

The fraction to be allocated is randomly chosen within a certain interval, because

Table 6.1: Model variables and algorithmic parameters

Model variables	
Sailing speed	20 knots
Ship capacity	12,000 TEU
Capital costs	\$11,350,000/yr
Operational costs	2,190,000/yr
Fuel costs	\$150/nm
Port costs	\$ 25,000/call
Handling costs	\$175/TEU
Revenue (westboud)	\$0.1677/nm/TEU
Revenue (eastbound)	\$0.0838/nm/TEU
Slack time	1 day
Port delay	0.5 days
Algorithmic parameters	
Number of service networks	100
Number of routes per network	6-15
Number of ports in a string	2-30
Fraction to randomly allocate	5%-15%

the fraction should not be too large or too small. A small fraction will result in tiny bits of demand being allocated, which is not reasonable, while a large fraction will interfere with the sorting techniques.

We choose the fraction to allocate by drawing a random number from the uniform distribution [5%,15%].

Table 6.1 shows a summary of all algorithmic parameters.

6.2.3 Data set

We use the Maersk Asia-Europe data set that is described in Chapter 5. It consists of a demand matrix with the yearly demand in TEU between 58 ports, a distance matrix containing the distances between these ports, a fleet of vessels, and a revenues and costs specification.

The data set is based on the actual Maersk Asia-Europe network during spring 2010. However, some estimations had to be made due to a lack of information regarding liner operations. The total demand on the trade lane is estimated at 6.78 million TEU. The distances between ports were obtained using online port-to-port distance calculators. The demand and distance matrix can be found in Appendix B. The actual network underlying the data set consists of 9 routes with an average length of 17 ports. To service these routes, Maersk operates a fleet of 91 vessels with a capacity ranging from 6,251 to 14,770 TEU, with the average capacity being approximately 8,500 TEU. Recall that we assume an unrestricted fleet of vessels, and as a consequence, do not use the fleet from the MAE data set. The revenue and cost data from the data set is already specified in Section 6.2.1.

Since the data set is based on an existing service network, we believe it represents reality. After we presented the results, we will compare our best network with the network underlying the data set. This enables us to put the performance of the algorithm into perspective, and assess the quality of the data set.

6.2.4 Implementation

The benchmark study consists of several simulations with different configurations of sorting methods and local search operators. These simulations are integrated in a single batch file, enabling easy collection of data. When running the presented experimental setup in our batch file, we obtain one large data file containing all simulation cycles. With the resulting data we can analyze the performance of the algorithm configurations with respect to effectiveness and efficiency. We also obtain the best network of services according to our algorithm, i.e., the solution with the highest profit from all our simulations. However, this might not really be the optimal solution.

For mathematical optimization problems in general, the use of heuristic algorithms does not guarantee finding an optimal solution. More often, only suboptimal solutions may be found, and to obtain the optimal solution, enumeration algorithms have to be used. The disadvantage of the exhaustive approaches is that they are extremely time-consuming. The difference between the solutions found using our multi-start local search algorithm and the optimal solution is called the optimality gap. The goal is always to reduce the gap as much as possible. In our case, it is difficult to judge the size of the optimality gap, since the optimal solution is not available. Obtaining it is practically impossible, which is a characteristic of a combinatorial optimization problem, like the one we face here. However, we try to estimate the gap in Section 6.3.3.

The simulation model is implemented in Matlab R2010a and is run on a Intel Core 2 Quad CPU at 2.40 GHz. Matlab is a numerical computation software package that allows for straightforward implementation of algorithms. It has many ready-to-use functions available, and we do not need additional libraries. We could have chosen for Java or C++, but these are less user-friendly and require more sophisticated language-specific knowledge. Actually, Matlab itself is written in Java and C++, but will not reach the performance of these languages. Therefore, the main disadvantage of Matlab is that it can be slow, depending on which programming constructs are used. Moreover, Matlab does not have code completion, as can be found in today's integrated development environments (IDE). In order to make the algorithm as fast as possible, we tried to use the most efficient programming constructs throughout the development process. For example, the use of an object-oriented approach as opposed to large vectors of data turned out to be much more efficient in Matlab. Besides, the object-oriented approach made modeling the problem much easier.

6.2.5 Hypotheses

Before we present the results of the benchmark study, we will review the hypotheses. In other words, what kind of results do we expect? In Section 1.3.1 we discussed the general hypotheses. Now that we have explained the algorithm, we can focus on our expectations in more detail.

The first hypothesis concerns the sorting method. The use of sorting methods to allocate demand to a network can improve the solutions. These methods however should be based on some rationale. The first method we have designed is quantity based, i.e. allocation starts with the origin-destination pair that has the largest demand. The second method is profit-driven, which means that we assign a three-attribute based score to each demand to indicate its profitability. The latter sorting method focuses on the allocation of profitable demands first, so that ship capacity is not 'wasted' with demands that are less profitable. Therefore, we expect that the profit-driven allocation (PDA) sorting method will perform better.

Hence, we expect that network profits after the insertion heuristic will be higher when the PDA sorting method is used. After the local search heuristic, which is the second phase of the algorithm, we expect that the networks with PDA Sort will still perform better than the networks with Quantity Sort, although a portion of the difference might be abolished by the local search heuristic.

The second hypothesis is related to the local search operators. We introduced three operators that try to improve solution candidates by exploring neighboring solutions. These operators are: the route-length operator which aims at adding new ports to a route and removing unprofitable ports, the port-exchange operator which tries to move ports within a route and between routes, and the transhipment operator, which focuses on the use of hubs and transhipment of cargo. Each operator contributes in its own way. The route-length operator is quite straightforward and directly contributes to the profitability when a port (and thus more demand) is added to a route, or when costs are saved by removing a port. The other two operators are less straightforward. The port-exchange operator can yield savings in port costs and fleet costs, but a large portion of this operator is dedicated to increasing capacity. Increased capacity is the aim of the *intra-route* part of the operator, while it can also be a side-effect of the *inter-route* part. Increased capacity has no direct effect on profitability, but it allows to allocate additional demand which generates new revenue. Increasing capacity is also the major goal of the transhipment operator, that first reroutes demand, after which additional demand can be allocated using transhipment.

Previously, we mentioned that we expect the transhipment operator to perform best, because transhipment is used in reality and should therefore substantially contribute to the objective function. However, there are other reasons as well. In Sections 3.2.3 and 6.2.1 we showed that fuel costs are a major cost driver, while all revenues have to come from the transportation of containers. This implies that a lot of money can be made when demand allocation is increased while fuel costs remain the same. The only way to obtain this is by increasing the average ship utilization in a network. When adding more demand is not feasible due to capacity constraints, one of the methods to increase utilization is to redistribute demand allocation in a network. Reallocating demand will cause some parts of the network to gain in capacity, while other parts will loose capacity. When this can be done such that new demand can be allocated, we increase the average utilization. This is the idea underlying the transhipment operator. Therefore, we expect that the transhipment operator will contribute to the objection function the most. After all, allocating new demand while costs stay more or less the same is the largest increase in profitability one can obtain.

Finally, we can express our expectations with regards to the final network layouts. There are a few characteristics that are related to both the data set and the algorithm, such as the average string length, the number of routes in a network, the fleet size, etc. When we look at the data set, we see that there are 58 ports. In the original network of Maersk there are 9 routes with 18 ports on average[1]. Since we have tried to model our demand, distance, cost, and revenue parameters close to reality, we expect that the

[1]The average of 18 ports is based on the number of visits, not the number of unique ports. Ports are allowed to be visited twice on the full trip.

final networks will not deviate from the Maersk network very much. Both the number of routes and the average string length seems feasible. The Maersk fleet on the Asia-Europe trade lane consists of 91 vessels with an average capacity of 8,500 TEU. Since we use 12,000 TEU vessels, we expect that our fleet will turn out a bit smaller.

6.3 Benchmark Results

In this section we discuss the results of the benchmark study. Two sorting methods and three local search operators yield 21 different configurations when we do not consider ordering. Table 6.2 shows the results of these 21 different configurations. We use P and Q to indicate the PDA Sort and the Quantity Sort method, respectively. Operators 1, 2 and 3 denote the route-length, port-exchange, and transhipment operator, respectively. Due to its design, we always apply the transhipment operator before the other local search operators, which explains the fact that operator 3 is always in the first position of a configuration[2].

The results in Table 6.2 show that configuration Q & 1-2 performs best, with a profit of \$4.81 billion. From the results, it also shows that the average profit across all 100 networks with this configuration is better than most other configurations. Only when we use the sorting methods P and P/Q, with the same operators 1 and 2, we obtain similar average profits. In fact, the three configurations where we use the operator combination 1-2 yield the highest profits. This already indicates that both sorting methods yield similar results, and our hypothesis that PDA Sort would outperform Quantity Sort seems not to hold.

To see whether there is any difference between Quantity Sort and PDA Sort, we analyze the network profits and compute the averages of both sorting methods. We do this both *before* the local search heuristic and *after* local search. Table 6.3 shows the average profits. These profits show that the difference is small, particularly after local search. For now, it seems save to assume that both sorting methods perform equally. In Section 6.4 we will perform statistical analysis to test this statement.

[2]This has to do with the negative performance that results from the first step of the transhipment operator. When the transhipment operator would be run after an other operator, there is less remaining demand in the demand matrix to allocate in the second step, in order to cover for the negative performance. Some more explanation regarding this behavior is given in Section 6.3.1. A more detailed discussion of the transhipment operator implementation is given in Section 7.2

[3]P denotes PDA Sort, while Q denotes Quantity Sort.

[4]1, 2 and 3 represent the 'route-length', 'port-exchange', and 'transhipment' operator respectively.

Table 6.2: Results from the benchmark simulation

sort method[3]	local search operator[4]	profit (average)	profit (best)	#routes (average)	#routes (best)	#ports (average)	#ports (best)	fleetsize (average)	fleetsize (best)
Q	1	$4.17bn	$4.72bn	10.2	8	22.9	26.2	87.8	71
Q	2	$2.52bn	$3.40bn	9.7	10	22.3	23.5	83.2	85
Q	3	$2.83bn	$3.79bn	10.1	8	22.4	25.1	87.5	71
Q	1-2	$4.24bn	**$4.81bn**	10.0	8	26.0	27.6	85.6	69
Q	3-1	$3.64bn	$4.28bn	9.7	8	24.6	29.0	83.1	70
Q	3-2	$2.92bn	$3.63bn	10.1	10	22.3	22.3	87.0	83
Q	3-1-2	$3.66bn	$4.36bn	9.6	8	24.3	28.3	82.5	71
P	1	$4.14bn	$4.66bn	9.8	8	26.0	29.6	83.4	73
P	2	$2.56bn	$3.74bn	10.1	14	21.9	22.8	85.7	118
P	3	$2.83bn	$3.59bn	10.2	10	22.0	20.7	87.2	86
P	1-2	$4.21bn	$4.74bn	10.2	8	25.4	30.3	86.4	72
P	3-1	$3.60bn	$4.35bn	10.1	7	24.0	29.0	86.1	64
P	3-2	$2.88bn	$3.56bn	9.7	9	22.2	23.1	83.8	78
P	3-1-2	$3.64bn	$4.29bn	9.9	8	24.0	29.0	84.1	74
P/Q	1	$4.17bn	$4.67bn	10.2	7	26.0	28.9	87.0	61
P/Q	2	$2.58bn	$3.63bn	9.9	11	22.0	23.6	84.6	94
P/Q	3	$2.89bn	$3.69bn	10.0	7	22.4	23.0	86.3	60
P/Q	1-2	$4.24bn	$4.74bn	9.8	8	26.0	30.4	84.3	69
P/Q	3-1	$3.56bn	$4.27bn	10.0	8	23.9	28.5	84.5	71
P/Q	3-2	$2.92bn	$3.81bn	9.7	7	22.4	21.9	83.7	59
P/Q	3-1-2	$3.50bn	$4.34bn	10.6	8	23.5	27.1	90.3	71

Table 6.3: Average profits with different sorting methods

	Before local search	After local search
Quantity Sort	\$2.45 billion	\$3.40 billion
PDA Sort	\$2.55 billion	\$3.42 billion

Table 6.4: Average improvement compared to phase I

Configuration	Route-length operator	Port-exchange operator	Transhipment operator
1	75.9%	-	-
2	-	3.0%	-
3	-	-	16.3%
1-2	75.9%	1.2%	-
3-1	28.8%	-	16.5%
3-2	-	0.4%	17.5%
3-1-2	28.2%	0.5%	17.0%
Average	**52.2%**	**1.3%**	**16.9%**

We are also interested in the performance of the local search heuristic. Therefore, we provide the individual and combined performances of each operator in Table 6.4. We aggregated the sorting methods because they hardly affect the performance of the operators, so the figures in the table are averaged across the three sorting method combinations. The last row represents the average increase of each operator, regardless of the configuration it was used in.

The 21 different combinations are the result of combining sorting methods and local search operators without considering ordering. Of course it is interesting to know if the sequence would affect the performance. However, if we add this relaxation to the complete benchmark, the number of possible combinations inflates to 60! Running a simulation of 60 combinations is computationally very intensive, so we chose to examine the impact of the sequence for one configuration only. For configuration P/Q & 3-1-2 we have run all different sequences of the local search operators, yielding 6 combinations in total. The results are presented in Table 6.5.

These results clearly indicate that the difference between alternative sequences is limited. Average profits are slightly lower when the transhipment operator is applied first, however, the sequence of operator 1 and 2 does not appear to affect the performance.

Figure 6.2 shows the profit distribution of all 2,100 networks (21 configurations, 100 networks each). The histogram indicates that there is a large spread between the networks. This is caused by the randomized multi-start heuristic, which sometimes gen-

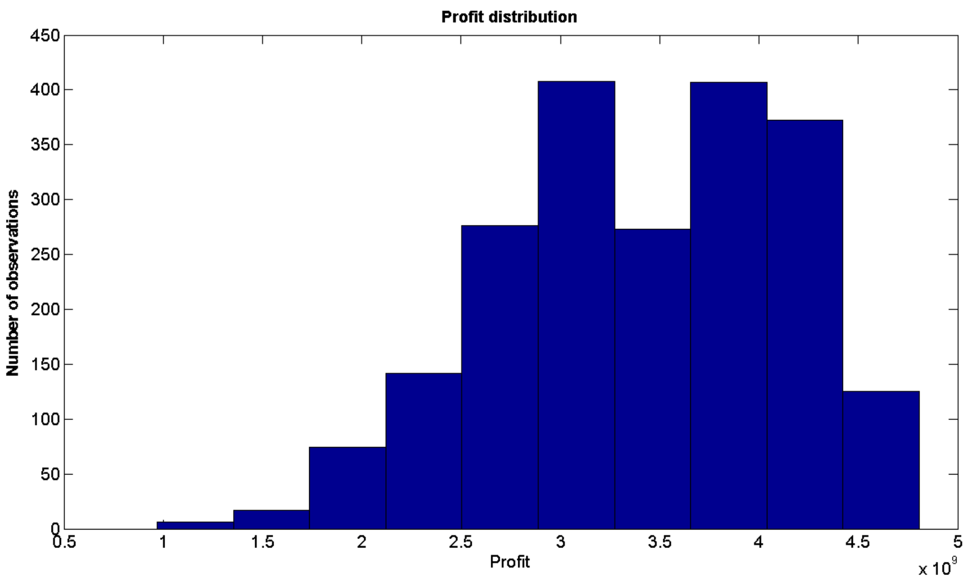


Figure 6.2: Profit distribution across all 2,100 networks

erates very poor networks that cannot be improved enough by the local search heuristic.

In Table 6.6, the average execution times for the different implementations of the two phases are presented. These numbers are based on the time needed to run the full phase, with the specified implementation, for 1 network. Note that here we do not consider multiple local search operators in phase 2. As a result, these numbers can be interpreted as the average time each operator would need when it is run in a configuration with another operator. However, in some cases the combined execution time is lower, as there is less to improve when another local search operator already improved the solution. Recall that these results are obtained using Matlab R2010a with a Intel Core 2 Quad CPU at 2.40 GHz.

Table 6.5: Various operator sequences for the PQ & 1-2-3 configuration

Sequence	Profit (average)	Profit (best)
1-2-3	\$3.77 billion	\$4.36 billion
1-3-2	\$3.83 billion	\$4.47 billion
2-1-3	\$3.71 billion	\$4.38 billion
2-3-1	\$3.68 billion	\$4.33 billion
3-1-2	\$3.50 billion	\$4.34 billion
3-2-1	\$3.55 billion	\$4.24 billion

Table 6.6: Average execution times of multi-start local search phases per network

	Execution time
Phase 1: Generating initial solutions	
Quantity sort	11.14s
PDA sort	11.54s
Phase 2: Local search optimization	
Route-length operator	17.82s
Port-exchange operator	8.30s
Transhipment operator	121.90s

For the first phase of the multi-start local search algorithm, we find that Quantity sort is slightly more efficient than PDA sort when it comes to execution time. This difference can be explained by the functioning of both sorting methods. The Quantity sort method sorts demands by their quantity, followed by a sorting of the feasible routes on remaining capacity. The PDA sort method does not sort the demands, but uses a more complex three-attribute based scoring system to sort the feasible routes. Still, the difference in execution time is very small since both methods spend most time on the other parts of phase 1, namely creating the routes and allocating the demand. Therefore, we can conclude that the real difference in execution time between the two sorting methods is negligible.

For the second phase, we find that the port-exchange operator is the most efficient. Perhaps this is related to its weak performance, which could mean it is not able to exchange many ports within and between routes, and as a result finishes early. The transhipment operator needs a lot of time compared to the other operators, since rerouting demands using transhipment yields much more route possibilities than the direct path approach of the route-length and port-exchange operator. Although the transhipment operator does not change the network, this does not compensate for the extra time needed to (re)allocate demands. Due to its high execution time, the transhipment operator might limit the number of networks to build when the algorithm is used within time constraints. The route-length operator has a reasonable execution time, especially when comparing its effectiveness with the other operators. It yields the most improvement per second execution time that is spent.

The execution times provided in Table 6.6 give an indication of the time needed to run a full algorithm configuration. For example, the multi-start local search algorithm Q & 1-2 configuration takes around 35 seconds to complete for one network. This comes down to approximately one hour when generating 100 networks. Based on this information, the Q & 2, P & 2, and P/Q & 2 configurations are the most efficient algorithms, all finishing within 20 seconds. In comparison with the most efficient networks, the most effective network (Q & 1-2) takes about 80% more time[5], but is approximately 68% more effective[6].

6.3.1 Interpretation

In Table 6.4 we can see that the route-length operator obtains around 75% profit increase when this operator is applied on its own, and it also does so when applied as first operator in a combination. Compared to the port-exchange and transhipment operator, the route-length operator performs best. Its success can be explained by its highly influential network adjustments and its profit-minded implementation. With respect to the other two operators, the route-length operator greatly influences the service network. Where the port-exchange operator only exchanges ports, the route-length operator actually adds and removes ports. This really changes the number of occurrences of a port in the service network, and thereby has far more influence on the final solution. The profit-driven approach that is used when making add-remove decisions makes sure the adjustments are only executed when the result yields a more profitable (or *effective*) network.

Whenever the route-length operator is run after the transhipment operator, its performance is substantially lower. Although the transhipment operator yields neat results of 16-17% profit increase, the combined performance of both operators is worse than the route-length operator alone. This is a result from the difficult cooperation between the two operators. The transhipment operator can allocate demand pairs for which no direct path exist, but that can be connected with the use of a transhipment hub. Transhipment paths are allocated over two routes, the origin-hub path is allocated in the first route, and the hub-destination path is allocated in the second route. When a network contains a large number of transhipments, it is difficult for the route-length operator to perform its task.

The route-length operator checks for each port in a route whether it is incurring more costs than revenue. Whenever this is the case, the port and its cargo allocations are removed from the route. However, the cargo allocations of that port might just contain a number of transhipment paths. It might be the origin, hub, or destination

[5]Based on an execution time of 19.44s for the Q & 2 configuration, and 35s for the Q & 1-2 configuration

[6]Based on an average profit of \$2.52b for the Q & 2 configuration, and \$4.24b for the Q & 1-2 configuration

of several transhipment paths that is removed. In this case, due to the removal of the transhipped cargo in one route, the demand allocations in the other routes are removed as well. After the route-length operator has removed ports, it tries to add ports based on the unfulfilled demand. However, the cargo that before was satisfied by transhipment is now more difficult to reallocate. It might just happen that very little of this demand gets allocated, yielding a substantial lower revenue than expected based on its single performance.

From Table 6.4, it also shows that the port-exchange operator has the worst performance of the three operators. It only produces around 1% profit increase. However, when applied after the route-length operator, its 1.2% profit increase is still better than no increase at all. The figures in Table 6.4 support the conclusion that the combination of operator 1 and 2 performs best.

The weak results of the port-exchange operator can be explained by carefully looking at the underlying ideas. On itself, exchanging ports within a route or between routes does not yield any financial advantages, since the amount of demand that is allocated remains the same. The only advantage is additional space on the ships, as a result of the port exchange and new paths for the demand pairs. After the exchange, revenue should be made by the allocation of new demand. However, a successful allocation depends on the remaining capacity on the ship. This is often a bottleneck during the allocation process. The results indicate that the additional space gathered was not enough to allocate all new demand, or there just was not so much demand to allocate.

Recall that in the final stages of creating the initial solution, as much demand as possible was allocated to the network. In order to successfully allocate a demand pair, a direct path between origin and destination is needed. After exchanging some ports, the number of new origin-destination direct paths is limited, especially when considering that our total service network consists of 58 ports. This is different for the transhipment operator, that is not bounded to direct paths, yielding more new allocation possibilities. Furthermore, a successful allocation needs enough space on its depicted path. Together, these two components keep the overall profit increase of the operator rather limited.

From reality perspective, one would expect that the introduction of transhipment would boost the profit of a network. Therefore, in our hypothesis we stated that the transhipment operator would outperform the other two operators. However, from the results it shows that the transhipment operator does not perform that good, especially in comparison with the route-length operator. In fact, the addition of the transhipment operator to another operator or a combination of the other two operators lowers the average profit. This can be explained by focusing on the two steps of the operator.

In the first step, the operator reallocates existing demand using transhipment paths whenever a shorter path over two routes can be reached. In this case, no new demands are allocated in the network, yielding a profit gain of 0%. The only gain is some extra space in the remaining capacities. However, we do incur extra handling cost for transhipping the demand from one route to another. As a consequence, this step of the transhipment operator yields a negative performance.

In the second step, the operator tries to allocate leftovers from the demand matrix. This is demand that could not be allocated because there was no direct path between origin and destination or there was not enough space. Using transhipment, the origin and destination are connected via a hub, yielding more path possibilities. The gain from this action is the extra demand that can be allocated, but the new allocations also incur handling costs. Overall, this part does yield a positive performance.

From the results it seems that the gain in the second step is rather moderate and, although it can compensate for the negative performance of the first step, it does not yield the desired increase. This can have the following reasons. It might be the case that there is not much demand left in the demand matrix to allocate. Another reason is that there is not enough space in the remaining capacities of the routes to successfully tranship demand. However, the most likely reason can be found in the chosen implementation of transhipment. Maybe too much demand is rerouted, incurring additional handling cost, while the space that is gained is not used for new allocations. Perhaps the current implementation of the transhipment operator is focused too much on the demand allocation, trying to increase revenue, while it neglects minimizing the cost side. The profits might have been higher when costs were cut by removing ports from a route while transhipping its demand to other routes. For a more detailed discussion of the transhipment operator implementation and its performance, we refer to Section 7.2.

6.3.2 Best network

In this section, we will present and discuss some detailed statistics of our best network. This gives us an idea how the network looks like. Additionally, we compare the network with the original Maersk service network that was used to construct the data set. In Table 6.7 we provide the details of the overall best network, which is the network with the Q & 1-2 configuration. From these figures, we can make the following observations.

Table 6.7: Detailed results of the best network (Q & 1-2 config.)

Config. execution time (sec)	3,813
Profit	\$4.81 billion
Demand fulfilled (TEU)	6,567,376 TEU
Demand fulfilled (%)	96.9%
Number of routes	8
Fleet size	69 vessels
Avg. ports per route	27.6
Distance traveled (nm)	222,842 nm
Number of paths	1641
Total revenue	\$7.57 billion
Total costs	\$2.76 billion
Avg. utilization per route	59.8%
Fully used legs (total)	3 (221)
OD pairs covered (total)	1641 (1934)

First, we see that the demand fulfilled is almost 100%. Although that is a very high score, it leaves us with the question: why is it not 100%? It could be that there is insufficient capacity to transport the remainder, or it may be that the last few demands are not profitable to fulfill. In the last row of the table we can see that not all OD pairs are covered. About 15% of the available OD pairs is not covered at all. It might be the case that there are no direct paths between the origin and destination of the remaining demand pairs. When we calculate the demand corresponding to those uncovered OD pairs, we find that the demand is 190,329 TEU, which is 2.8% of the total demand. We can already sense that the demands of the uncovered OD pairs are quite small, because those 15% of the OD pairs make up for only 2.8% demand. Anyway, this leads to the conclusion that the lack of direct paths is responsible for most of the unallocated demand. The other 0.3% of the unallocated demand is probably not fulfilled because of capacity constraints and unprofitable amounts of demand between OD pairs.

Second, we observe that the utilization rate is almost 60%, this seems to be rather low. However, recall that there is a substantial trade imbalance between the eastbound and westbound direction on the Asia-Europe trade lane. More specifically, the westbound direction is responsible for 71.97% of the total demand, and the eastbound direction for the remaining 28.03%. With this information, we are able to compute the maximum attainable average utilization rate on the trade lane, which equals (71.97 + 28.03)/(2 * 71.97) = 69.47%. Given this value, our average utilization per route of 59.8% is not even that bad.

Still, the utilization rate implies that we could use smaller ships than the 12,000 TEU capacity ships that we are currently using. However, when taking a closer look at the utilization on each route, we find that only one of the eight routes has an over-capacity of more than 1,000 TEU. On this route we could actually use a 8,000 TEU ship. All other routes have at least one or more legs that require nearly full capacity. This leads to the discussion: what ship size would be optimal to use?

A full study on optimal ship size is beyond the scope of this study, however, we did some research. In an additional experiment, we used a ship capacity of 8,000 TEU, which is about the average ship size that Maersk uses on the Asia-Europe trade lane. We tried to run a simulation of 100 networks using the Q & 1-2 configuration, with the smaller ship size. In Section 7.3 we provide the results of this experiment. It turned out that the overall performance is worse. The best network has a profit that is approximately 6% lower. Also, the utilization rate is significantly lower. This leads to the conclusion that smaller ship sizes do not guarantee higher utilization, let alone higher profitability.

In the same additional experiment, we also tested a mixed fleet. By removing cargoes we optimized the ship size per route, leading to a fleet with different ship sizes. In contrast to the fixed ship size of 8,000 TEU, this approach did yield profitable results, and a new best solution was found. The average utilization rate across 100 networks increased from 45.4% to 59.6% after introducing a mixed fleet. An extensive description of this additional experiment is given in Section 7.3.

Third, when we focus on the network structure, we see that the best network consists of 8 different routes. This corresponds more or less to the original Maersk data set, which has 9 routes. In Appendix C we provide an overview of all routes and port occurrences in our network. When we compare these routes with the original Maersk routes from Appendix B, we can see that our routes, on average, are longer. The average route length in the original Maersk network was 16 ports, against 27.6 for our network. This can have the following reasons.

For the first reason, recall from Chapter 4 that the route-length operator is the only operator influencing the number of ports in a route. In our algorithmic parameters, we allowed a maximum string length of 30. However, this boundary only applies for the network generated by the randomized initialization phase. The route-length local search operator can extend a route even further outside this boundary. Maybe the difference in average route length could have been limited by lowering the boundary of 30 ports.

Another reason that may play an even more important role regarding the high string length is the lack of transhipment. Since our best network does not incorporate transhipment, all demand pairs have to be satisfied by direct paths between origin and destination. This means that the origin and destination of a demand pair should occur in the same route, in order to allocate its demand. Suppose we have a network and some remaining demand pairs to be allocated, of which no direct paths exists in the current network. When we apply the (inter)port-exchange operator, ports are exchanged between routes, and new direct path possibilities to satisfy our demand pairs might be created. However, the exchange of ports also removes already allocated demands. The only way to satisfy additional demand while keeping the existing demand allocations, is to use the route-length operator. As a result, ports are added to the routes as long as the demand that can be allocated is enough to cover for the additional costs. In this case, it seems that it was profitable to extend the routes with quite some ports, in order to obtain the additional demand allocations.

Fourth, the fleet of our network is smaller than the original Maersk fleet from the data set. Our network needs 69 vessels against 91 vessels for the Maersk network. However, there are substantial differences in the vessel size used. Maersk operates a fleet with a capacity ranging from 6,251 to 14,770 TEU, with the average capacity being approximately 8,500 TEU. Recall that we assume an unrestricted fleet of vessels with a capacity of 12,000 TEU. This makes the specific number of vessels incomparable, but we can still compare the total capacity used on both networks.

Using Appendix Table B.4 we find that the total capacity on the Maersk network is 768,559 TEU. The total capacity on our network is $69 * 12,000 = 828{,}000$ TEU. The difference is not that large, especially when considering our average utilization, which we expect to be somewhat lower than the average utilization on the ships in the original Maersk network. Unfortunately, we have no information regarding the utilization in Maersk's network.

The financial statistics of the best network yield excellent values, but can be point of discussion. Particularly the large difference between total revenue and costs, where the revenue is almost a factor 2.75 larger than the costs, is striking. We obtained a profit of \$4.81b with our best network. When looking at the Maersk 2010 interim report, we find a total profit of \$0.49b in the first half of the year on the Asia-Europe trade lane. For the full year, a total profit of around \$1b can be expected. This means our network generated almost five times as much profit as the original Maersk service network, even though the data set was based on that network. This rather large deviation can have several causes.

First, the revenue of the Maersk network should be corrected for the CMA CGM vessels that operate in a joint operation. The demand that these vessels take care of is incorporated in the demand matrix, but its revenue is not incorporated in the numbers from the interim report. The CMA CGM vessels are responsible for 9,5% of the demand in the demand matrix, such that we can increase the profit of the Maersk network to \$1,095b. Naturally, this increase does not help much, the difference remains substantial.

Second, there might be inaccuracies in the model variables, especially regarding the revenue and cost variables. The large difference between total revenue and costs for our best network might indicate unrealistic values for the revenues, the costs, or both. The revenue variables, split in westbound and eastbound revenue, are determined based on a historical average, and should be representable. The cost variables might be of more importance in this respect. Actually, the chosen cost structure might be too limited, and not take all specific costs into account that Maersk does. For example, our study does not incorporate overhead costs, only the costs directly following from the operation of the liner ship. Furthermore, for the fuel cost, we based our estimation on other studies. Since the fuel price is very volatile, the fuel costs might be estimated too low. The profit of a network is very sensitive to its revenue and cost data, such that a small deviation from reality can have large influence on its value.

Figure 6.3 represents one of the routes in our best network. The figure shows the loading and offloading activities at each port, as well as the ship utilization between port calls. The utilization rates clearly show the problem that shippers face in real life. On some parts of the route, almost full capacity is reached, while utilization drops below 40% on other parts of the routes. The difference between the eastbound and westbound direction becomes very clear when looking at the sample route. The turning ports in this route are Kwangyang and Gdansk. Note that the vessels load more cargo in Asia than they offload, and that the vessels offload more in Europe than they load. This indicates the trade imbalance[7] on the Asia-Europe trade lane. In line with the loading and offloading movements goes the utilization of the ships. The utilization is at its best (95.1%) when leaving Asia.

[7]Westbound demand is responsible for 71.97% of the total demand.

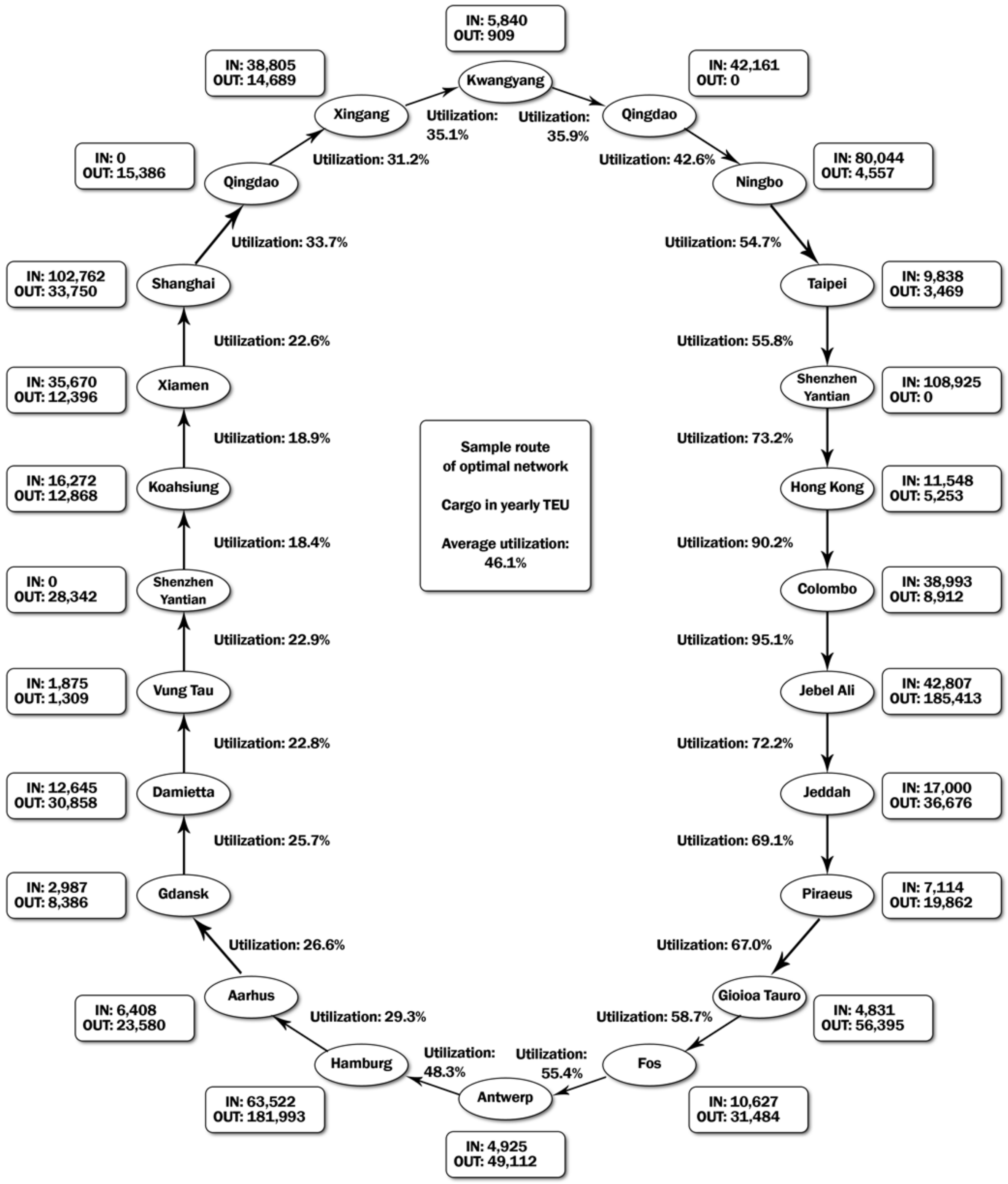


Figure 6.3: Sample route of the best network

6.3.3 Optimality & Upper bound

In the previous section we provided a comparison of our best network with the original Maersk service network that the data set is based on. We have seen that the profit from our best network is more than four times the actual profit of the Maersk network. We tried to justify this difference, but it is clear that our assumptions and variable settings do not perfectly match the ones from Maersk. As a result, it is difficult to make the comparison with the original Maersk profit. However, there is another way to place the performance of our algorithm into perspective, namely by comparing our best profit with the *upper bound*.

The upper bound is a theoretical bound of the highest possible profit that can be obtained, based on the particular assumptions we made and the variable settings used. Note that this differs from the optimal solution, since the upper bound is no actual solution, it is merely an indication of the maximum attainable profit given the variable settings. In our case, the upper bound does not even create a service network neither is it based on one. The upper bound profit is the result of fulfilling all demand, and only incurring the costs that are directly related to satisfying this demand. In general, the optimal solution will not be able to reach the profit of the upper bound.

The upper bound is computed in the following way. We use exactly the same model variable settings as presented in Section 6.2.1, apart from the slack time, which is set to zero. The rationale behind this decision is that the upper bound does not use a network, such that slack does not occur. Furthermore, recall that the upper bound does not assume a weekly service. In order to obtain the upper bound profit we need to compute the maximum attainable revenue, based on transporting all demands in the demand matrix, and the accompanying costs.

We walk through the demand matrix, and for each demand pair we compute the revenue and its costs. The revenue follows from multiplying the OD demand by the shipping rate, either eastbound or westbound, and the distance in nautical miles. Since we are computing the upper bound, we want minimum costs for transporting the demands. To that end, we compute the number of ships needed for transporting the OD demand by dividing the yearly OD demand by the vessel capacity (12,000 TEU) while assuming 100% utilization. We allow a non-integer number of ships, such that costs are only assigned to the ship capacity that was actually used. For example, if the cargo between port A and B is 3,600 TEU, we can multiply all costs, except for the handling cost, with 30%[8]. In addition, we compute the number of days needed for cruising from origin to destination, such that the yearly capital and operational costs can also be discounted to the number of days the ship was actually used for the specific demand pair. In this way, we compute the total cost of each demand pair, consisting of capital and operational costs, fuel costs, port costs, and handling costs. Now that we calculated the total revenue and the total costs, one might think we are able to compute the upper bound profit.

However, we still have to cover for the trade imbalance between the eastbound and westbound direction of the Asia-Europe trade lane. Up to this point, we considered origin-destination pairs separately, that is, we did not consider A-B and B-A at the same time. In order to cover for the trade imbalance between A-B and B-A we have to add additional costs, since the number of ships needed, based on the 12,000 TEU ship size, on the east- and westbound directions differ. Naturally, it makes no sense to cruise with a different size on the westbound direction as one would on the eastbound direction. Therefore, we have to compute the costs of the imbalance, and its resulting empty slots, for each full demand pair (e.g., A-B and B-A). Again we walk through the demand matrix, but now we consider the full demand pair at once. For each full demand pair, we compute the trade imbalance. Using this value, we again compute the fraction of the 12,000 TEU ship that we need to satisfy this imbalance. Then, we compute the additional capital and operational costs, fuel costs, and port costs, in a way analogous to the cost calculation above. Note that there are no additional handling costs, since the costs for handling the containers are already taken into account in the previous calculation. Now that we have computed the total additional costs for all full demand pairs, we add this number to the total costs that we found in the previous calculation. Finally, we are able to determine the upper bound profit by subtracting the total costs from the total revenue. The upper bound revenue is \$7.79b and the costs are \$2.70b, yielding an upper bound profit of \$5.09b. The imbalance costs are responsible for \$469m of the total costs.

An overview of the upper bound and our best solution is given in Table 6.8, together with the absolute difference between them. The upper bound of 5.09 billion dollar is approximately 280 million dollar higher than the profit of our best solution. This places the performance of the multi-start local search algorithm into perspective. In Section 6.2.4, we argued it is impossible for us to determine the optimality gap, the difference between our solution and the optimal solution. However, now we do have

[8]The value of 30% is obtained by computing 3,600/12,000.

information on the difference between our solution and the upper bound. This relative gap equals 5.5%[9].

Table 6.8: Performance of best network versus upper bound

	Best network	Upper bound	Δ
Profit	4.81b	5.09b	0.28b
Revenue	7.57b	7.79b	0.22b
Costs	2.76b	2.70b	-0.06b

From Table 6.8 we can see that the revenue for the upper bound is $0.22b higher than the revenue from our solution. This difference directly follows from the amount of demand that is fulfilled. Where the upper bound satisfies all demand, our solution only satisfies 96.9% of the total demand. Although the revenue of the upper bound is higher, its costs are still lower than the costs incurred by our solution. Recall that the upper bound is very efficient with its 100% utilization of ships and does not incur costs that follow from having a network, like extended cycle duration because of additional slack time. Also, in the upper bound we calculate the costs for transporting each demand pair separately, assuming we shuttle between origin and destination, and we discount the costs to the exact ship volume that is used. Since our best solution does have a network with weekly services, its utilization is considerably lower, resulting in a larger number of ships needed to transport the demand. This ultimately results in the $0.06b difference in costs.

Note again that the upper bound does not resemble the optimal solution. The optimal solution will actually be lower than the upper bound, which is a theoretical bound based on perfect circumstances. Given this information, we believe the algorithm performs well for a heuristic approach.

6.4 Result Analysis

In Section 6.3 we have provided the results of the benchmark study. The raw figures are not truly meaningful without a proper analysis. In this section, we use some basic methods to analyze the results and give an interpretation of the outcomes. This enables us to answer the subquestions that we formulated in the first paragraph of this chapter.

[9]The relative gap is defined as the absolute gap divided by the upper bound.

6.4.1 Analysis design

In this section we are eager to answer the research subquestions which relate to the performance of the algorithm, such as: *what local search operators contribute the most to the objective function?*, and *what is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping?*. In order to answer these questions, we need to carefully analyze the results we have provided in Section 6.3. Raw figures are not conclusive as to what method performs better than the other, unless we perform some sort of statistical analysis. The main reason is that we use random numbers, so that we cannot control the variation. Because different configurations are not based on the same set of random numbers, we need to perform a statistical analysis.

We are also interested in the composition of profitable networks as opposed to less profitable networks. Can we find a relation between, for example, the number of routes in a network and its profitability? In the following paragraphs we will explain what methods we use to analyze the results.

Statistical analysis

The benchmark study provides results for each combination of sorting method and local search operator. We would like to assess whether one result is significantly better than the other, or not. To test this, we subject the results to a *Student's t-test*. We use the unpaired t-test, which can test the null hypothesis that the population means related to two independent, random samples from an approximately normal distribution are equal. The unpaired t-test, which is formulated in equation 6.3, is defined by Armitage & Berry (1994).

$$\begin{aligned} t &= \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{s^2(\frac{1}{n_1} + \frac{1}{n_2})}} \\ s^2 &= \frac{\sum_{j=1}^{n_1}(x_j - \bar{x}_1)^2 + \sum_{i=1}^{n_2}(x_j - \bar{x}_2)^2}{n_1 + n_2 - 2} \end{aligned} \tag{6.3}$$

where $\bar{X}_1$ and $\bar{X}_2$ are the samples means, s^2 is the combined sample variance, and n_1 and n_2 are the samples sizes.

If there is a significant difference in the variances of the two samples, the unpaired t-test is not appropriate. We should then use a slightly different formulation of the unpaired t-test, which is known as *Welch's t-test* (Armitage & Berry 1994) and is formulated in equation 6.4.

$$
\begin{aligned}
t &= \frac{\bar{X}_1 - \bar{X}_2}{s_{\bar{X}_1 - \bar{X}_2}} \\
s_{\bar{X}_1 - \bar{X}_2} &= \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \\
\text{where } s_k^2 &= \frac{\sum_{j=1}^{n_k} (x_j - \bar{X}_k)^2}{n_k - 1} \quad , k \in \{1, 2\}
\end{aligned}
\tag{6.4}
$$

For each pair of samples, we have to assess whether there is a significant difference in the sample variances. We can test this using a two-sample F-test. In this test, the null hypothesis H_0 assumes that the variances are equal, and the alternative hypothesis H_1 assumes that they are different. We calculate the F-ratio by dividing the larger sample variance by the smaller sample variance. We can compare that F-ratio with $F_{\alpha/2}$, which can be found in the F-distribution table. We would also need the degrees of freedom: $df_1 = n_1 - 1$ and $df_2 = n_2 - 1$. We reject H_0 if $F > F_{\alpha/2}$, in which case we have to use the Welch's t-test because the two sample variances are not equal. Otherwise, we can assume the variances are equal and we use the unpaired t-test. The steps of the F-test are shown in table 6.9.

Table 6.9: Using the F-test

hypothesis:	$H_0 : \sigma_1 = \sigma_2$ $H_1 : \sigma_1 \neq \sigma_2$
test statistic:	$F = \frac{\text{larger sample variance}}{\text{smaller sample variance}}$
deg. of freedom:	$df_1 = n_1 - 1$ $df_2 = n_2 - 1$
rejection:	reject H_0 if $F > F_{\alpha/2}$

Using the basic methods we provided above, we can assess whether one result is significantly better than the other. This leads to answering the qualitative subquestions we have formulated earlier.

Network composition analysis

Besides the performance of the algorithm and its components, we are also interested in the question: what does a good network look like? We have collected data about the composition of networks during the benchmark simulation, so that we can analyze these data and, hopefully, can identify relationships between a network's composition and its profitability. This will also help to better understand the algorithm, and avoid the image of a 'black box' where some magical operations produce profitable networks without any intuition. We might expect that, for example, the number of routes in a profitable network is higher than those in less profitable ones. Although it is by definition not the case that a large bunch of routes make a good network, we can say that neither too few or too many routes are helpful towards a profitable network. The same issue applies to the average number of ports in a route. We hope that a thorough analysis of the network's composition will lead to the answer to these questions. In particular, we will use data of the following statistics for each network:

1. number of routes
2. average number of ports per route
3. average route utilization (eastbound)
4. average route utilization (westbound)
5. number of operated vessels
6. percentage of containers that is transhipped
7. percentage unallocated containers

We use multiple linear regression (MLR) to identify relationships between these characteristics and the profitability of a network. MLR is a simple and straightforward technique to examine the linear correlations between independent variables (defined above), and a single dependent variable, in this case *profit*. An MLR analysis produces regression coefficients (β's) which indicate to what extend the *profit* is predicted by each of the independent variables.

A typical MLR model looks like the formula in equation 6.5.

$$\begin{array}{l} y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon \\ \quad \text{where } y \text{ denotes the dependent variable (profit), } x_1 \text{ ... } x_k \\ \quad \text{denote the independent variables (features), } \beta_0 \text{ denotes the} \\ \quad \text{constant term, } \beta_1 \text{ ... } \beta_k \text{ denote the regression coefficients,} \\ \quad \text{and } \epsilon \text{ is the error term.} \end{array} \tag{6.5}$$

For each network we can obtain these regression coefficients, which tell us how strong the relationship is between the characteristics of a network and its profitability. If the coefficient of one feature is larger than that of another feature, its relationship is stronger.

6.4.2 Analysis

We start with the statistical analysis, which we can apply to different result data. First, we will test whether the 'winning' configuration really performs better than the second-best configuration, or third-best configuration. Second, we will compare the two sorting methods and test whether there is a significant difference between Quantity Sort and PDA Sort, both for the initial profits (before local search) and final profits (after local search). Third, we will further analyze the impact of operator sequence. We showed the results for different sequences of all three operators, and will test if the differences are significant.

Comparing configurations

The configuration Q & 1-2 provided the network with the highest profit. Also, the average profit of this configuration is the highest, together with the P & 1-2 configuration. But the average profit of the Q & 1 configuration is very close, only 1.7% lower than the Q & 1-2 configuration. To test whether the best network performs significantly better than other configuration, we apply a so-called t-test. Using this test, we compare the profits of the configurations that have an average profit larger than \$4 billion.

First we have to assess whether there is a significant difference in the variances of the samples, using an F-test. In Section 6.4.1 we explained how to use the F-test. In Table 6.10 we provide the F-ratios for each combination of configuration.

The table shows the F-ratio for all combination pairs, which is equal to the larger sample variance divided by the smaller sample variance. We reject the null hypothesis

Table 6.10: Different configurations: F-ratios

	Q & 1	Q & 1-2	P & 1	P & 1-2	P/Q & 1	P/Q & 1-2
Q & 1	1.00	1.01	1.19	1.25	1.31	1.11
Q & 1-2	1.01	1.00	1.18	1.24	1.32	1.10
P & 1	1.19	1.18	1.00	1.05	**1.57**	1.08
P & 1-2	1.25	1.24	1.05	1.00	**1.65**	1.13
P/Q & 1	1.31	1.32	**1.57**	**1.65**	1.00	1.46
P/Q & 1-2	1.11	1.10	1.08	1.13	1.46	1.00

(two sample variances are equal) when the F-ratio is larger than $F_{a/2}$. A common value for alpha is 0.05, so we will use that value as well. Using $df_1 = df_2 = 100 - 1 = 99$, we obtain $F_{0.025} \approx 1.52$ from the F-distribution table. Hence, for all combinations with an F-ratio smaller than 1.52 we assume that the variances are equal, otherwise they are not. In Table 6.10 we marked the values for which the pairs have unequal variances bold.

Now that we know for which configuration pairs the variances are equal and for which they are not, we can perform the corresponding t-test. Samples with equal variances are subjected to the unpaired t-test, while the Welch's t-test is used for samples with unequal variances.

We start with the first pair: configurations Q & 1 and Q & 1-2. We know that their variances are equal, so we use the unpaired t-test. First we calculate the combined sample variance s^2:

$$s^2 = \frac{\sum_{j=1}^{n_1}(x_j - \bar{x}_1)^2 + \sum_{i=1}^{n_2}(x_j - \bar{x}_2)^2}{n_1 + n_2 - 2} = \frac{(5.9524 + 5.9949) \cdot 10^{18}}{100 + 100 - 2} = 60.3399 \cdot 10^{15} \tag{6.6}$$

Next we compute the value for t:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{s^2(\frac{1}{n_1} + \frac{1}{n_2})}} = \frac{(4.2423 - 4.1679) \cdot 10^9}{\sqrt{60.3399 \cdot 10^{15} \cdot (\frac{1}{100} + \frac{1}{100})}} = 2.141 \tag{6.7}$$

Using the t-value, we can obtain the p-value from the table of values from the Student's t-distribution. The t-value of 2.141 corresponds to a p-value of 0.034. If the p-value is below the statistical significance threshold α, then the null hypothesis (sample means are equal) is rejected. We use the common threshold $\alpha = 0.05$, which leads to the conclusion that the p-value is smaller than α, and the null hypothesis is rejected. Hence, the sample means are not equal and we can safely assume that the Q & 1-2 configuration

Table 6.11: Different configurations: p-values

	Q & 1	Q & 1-2	P & 1	P & 1-2	P/Q & 1	P/Q & 1-2
Q & 1	1.000	0.034	0.466	0.305	0.980	0.035
Q & 1-2	0.034	1.000	0.006	0.323	0.025	0.975
P & 1	0.466	0.006	1.000	0.095	0.427	0.007
P & 1-2	0.305	0.323	0.095	1.000	0.288	0.319
P/Q & 1	0.980	0.025	0.427	0.288	1.000	0.027
P/Q & 1-2	0.035	0.975	0.007	0.319	0.027	1.000

performs significantly better than the Q & 1 configuration.

Similarly, we compute the p-values for all other configuration pairs. In Table 6.11 we show these p-values.

Recall that we reject the null hypothesis, the assumption that sample means are equal, if a p-value is smaller than $\alpha = 0.05$. Using the table's values, we can see that the best configuration, Q & 1-2, performs significantly better than Q & 1, P & 1 and P/Q & 1, but performs equally as P & 1-2 and P/Q & 1-2. Hence, although their average profits are close to the best configuration, the configurations where only the route-length operator is used perform significantly worse than the use of both the route-length and the port-exchange operator. This supports the general conclusion that the use of the operators 1 and 2 leads to the best results.

It also seems that the use of the sorting method is not really relevant. This leads us to the next statistical test: is there a significant difference in the use of different sorting methods?

Comparing sorting methods

We introduced two different sorting methods that are used to allocate demands to initial networks. We would like to know whether one method performs better than the other. The results of the configurations are not conclusive, so we have to analyze the different results of both sorting methods separately. When we aggregate all 21 configurations and analyze the profits, we can compute the average profit of networks that have used the PDA Sort method, and of networks that have used the Quantity Sort method. We do this both before local search and after local search. The profits before local search will tell the most about the performance of each sorting method, but of course, we are mainly interested in the final results (after local search). In Table 6.3 we presented the average profits. Before local search, Quantity Sort performs on average 4% better, while

Table 6.12: Two sorting methods: F-ratios

	Q before local search	P before local search	Q after local search	P after local search
Q before local search	1.00	1.01		
P before local search	1.01	1.00		
Q after local search			1.00	1.01
P after local search			1.01	1.00

Table 6.13: Different configurations: p-values

	Q before local search	P before local search	Q after local search	P after local search
Q before local search	1.000	0.000		
P before local search	0.000	1.000		
Q after local search			1.000	0.536
P after local search			0.536	1.000

the difference is only 0.6% after local search. To assess whether these differences are significant, we perform a statistical analysis.

Similar to the configuration analysis above, we compute the F-ratios and compare them with $F_{\alpha/2}$. The degrees of freedom (sample sizes minus one) for Quantity Sort and PDA Sort are 1,068 and 1,030 respectively, so that we obtain $F_{0.025} = 1.11$. Table 6.12 shows the F-ratios. All ratios are smaller than 1.11, so that we assume that sample variances are equal.

Next, we subject the samples to the unpaired t-test. In Table 6.13 we present the p-values. Recall that p-values lower than $\alpha = 0.05$ indicate that sample means are not equal, which leads to the conclusion that one method performs significantly better than the other.

The p-values indicate that the Quantity Sort method performs significantly better than the PDA Sort method when we compare them before local search, but after local search there is no significant difference between the two sorting methods.

Comparing local search operator sequence

Local search operators can be applied in random order. However, the operators have a direct effect on the structure of the networks. Ports might be moved, added, removed, etc. The work of one operator affects other operators. Therefore, we have tested the performance of different sequences. Three operators can be applied in six different sequences. We tested these sequences using the PQ sorting method. The average profits are listed in Table 6.5. From this table, it appears that the operator sequence 1-3-2 performs best, with an average profit of \$3.83 billion. But the second best, sequence 1-2-3, has an average profit of \$3.77 billion. Is the best sequence significantly better than the second best sequence? To answer this question, we subject the samples to a t-test again.

Table 6.14: Different operator sequences for P/Q: F-ratios

	1-2-3	*1-3-2*	*2-1-3*	*2-3-1*	*3-1-2*	*3-2-1*
1-2-3	1.00	1.02	1.02	1.30	1.17	1.13
1-3-2	1.02	1.00	1.00	1.28	1.19	1.15
2-1-3	1.02	1.00	1.00	1.27	1.19	1.15
2-3-1	1.30	1.28	1.27	1.00	1.52	1.47
3-1-2	1.17	1.19	1.19	1.52	1.00	1.03
3-2-1	1.13	1.14	1.15	1.47	1.03	1.00

First, we calculate the F-ratios and compare them with $F_{0.025} \approx 1.52$. For sample pairs with an F-ratio smaller than 1.52 we assume that their sample variances are equal and apply the unpaired t-test. From the F-ratios provided in Table 6.14 we conclude that most sequences have similar variances, only sequences 3-1-2 and 2-3-1 have an F-ratio that is around the value of 1.52. Due to rounding, it is not obvious from the table, but the F-ratios actually exceed $F_{0.025}$, so that their sample variances are not equal.

Next, we apply the appropriate t-test to each sequence pair. We apply the unpaired t-test for all pairs, except for the sequence combination 3-1-2 and 2-3-1, for which we use Welch's t-test. The corresponding p-values are given in Table 6.15.

We conclude that one sequence performs better than another if their p-value is smaller than $\alpha = 0.05$. When we look at average profits, Table 6.5 shows that sequence 1-3-2 performs best, followed by 1-2-3. However, their p-value is 0.263 which is substantially larger than 0.05. Therefore, we cannot say that there is a significant difference between the performance of the sequences 1-2-3 and 1-3-2. When we compare the best sequence (1-3-2) with the third-best sequence (2-1-3), we find a p-value of 0.023. Because that value is smaller than 0.05 we reject the null-hypothesis that sample means are equal,

Table 6.15: Different operator sequences for P/Q: p-values

	1-2-3	*1-3-2*	*2-1-3*	*2-3-1*	*3-1-2*	*3-2-1*
1-2-3	1.000	0.263	0.248	0.045	0.009	0.000
1-3-2	0.263	1.000	0.023	0.001	0.000	0.000
2-1-3	0.248	0.023	1.000	0.426	0.125	0.002
2-3-1	0.045	0.001	0.426	1.000	0.388	0.013
3-1-2	0.009	0.000	0.125	0.388	1.000	0.141
3-2-1	0.000	0.000	0.002	0.013	0.141	1.000

and conclude that the best sequence performs significantly better than the third best. This shows that, although the differences are small, it is best to run operator 1 first, before applying the other two operators.

Network Composition Analysis

In this part, we analyze the impact of a network's composition and properties on its profitability. To this end, we select a set of features that describes each network and perform multiple linear regression (MLR). The regression analysis can tell what features have a strong impact on the profitability of a network. For example, if profitable networks tend to have long routes, the MLR analysis will turn up with a high regression coefficient for that particular feature.

In the first step, we prepare the data for the MLR analysis. This means that we compose a table with a network on each row, and all the features in the columns. In the last column we place the profit of each network, which will be the dependent variable.

Before we apply the regression model, we normalize all feature data to obtain values between zero and one. We do this by using the min-max normalization, which subtracts the minimum value of a feature from each of its values, and then divides the difference by the range of the feature. Although this procedure is not necessary for the regression model itself, it will be easier to interpret the regression coefficients if data is normalized, because it allows direct comparison between coefficients.

The results of the regression model are provided in Table 6.16. We have listed all the features, including their minimum, maximum, and mean values. The last column shows the regression coefficients for the features and the constant term. The regression analysis also showed that $R^2 = 95.9\%$, which indicates that the model has a proper fit.

The regression coefficients lead to some important conclusions regarding the rela-

Table 6.16: MLR: overview of features and their regression coefficients

	min	*max*	*mean*	β (x 10^9)
Constant term				4.83
Number of routes	6	14	9.9	-0.99
Average number of ports per route	17.7	33.6	24.2	-0.29
Average route utilization (eastbound)	9.3%	64.5%	28.3%	-1.08
Average route utilization (westbound)	21.3%	88.3%	47.7%	1.77
Number of operated vessels	9.4	11.8	10.6	-0.26
Percentage of containers that is transhipped	0	60.0%	22.3%	-0.73
Percentage unallocated containers	0.4%	65.0%	14.0%	-3.49

tionships between a feature and a network's profitability.

First, there is a strong negative relationship between the percentage of unallocated containers and the profitability. In other words, allocation of containers leads to a higher profit. In itself, this conclusion is rather obvious. However, the coefficients show that this feature is the most important one. Not the route utilization, nor the size of a network is the most important, but how much of the available demand you can allocate is crucial. This is also important for future reference, because it indicates that profit will benefit from improving allocation the most.

Second, we see that both the number of ports per route and the number of operated vessels are of minor relevance compared to the other features. Their coefficients are rather small, only around -0.25. This might be explained by the fact that both port costs and vessel costs are not that important. It is far more important what routes are used, and how much demand can be allocated to them.

Third, the coefficient for the number of transhipped containers is negative. This is not surprising, given our findings in Section 6.3. In that section, we concluded that the transhipment operator has a poor performance. Transhipping containers is rather expensive, and it appears to be difficult to allocate enough extra demand to compensate for those additional costs. Again, the regression coefficient of $\beta = -0.73$ supports the conclusion that the algorithm performs better without transhipment.

Finally, we find that the westbound route utilization is positively related with profit, while the eastbound route utilization is negatively related. This appears quite strange, because one would expect that a higher utilization, either eastbound or westbound, would yield a higher profit.

The fact that we encounter an unexpected sign triggers warning signals. When we think about these two utilization factors more carefully, we feel that there must be a strong dependency between them. After all, route utilization will be higher when a network is properly designed, i.e., when the routes and the ports on those routes allow for a large allocation rate. Also, utilization will be limited when the remaining capacity on a route is insufficient. This is likely to affect both eastbound and westbound allocation.

To test this theory, we calculate the correlation between the factors. A correlation coefficient $\rho = 1$ means perfect dependency. When we calculate the correlation between the eastbound and westbound utilization, we find that $\rho = 0.96$. In comparison, most correlation coefficients between the other features do not exceed $\rho = 0.5$. This supports the idea that eastbound and westbound utilization are strongly related.

The problem with correlated variables in a regression model is that they can cause what is called *multicollinearity*. This phenomenon occurs when two or more variables are strongly related. Multicollinearity can affect the coefficients of those related variables severely. While the model as a whole might be accurate (or *fit*), one should not trust the individual coefficients of correlated features. The problem is explained in more detail by Farrar & Glauber (1967).

Multicollinearity can be solved in multiple ways, one of which is to exclude either one of the correlated features, or to combine them into one feature. We will suffice with a short analysis with either one of the features left out. When we leave out eastbound utilization, the regression coefficient for the westbound utilization becomes $\beta = 0.67$. Vice versa, for eastbound utilization we find that $\beta = 0.22$. All other coefficients are more or less the same as listed in Table 6.16.

These results seem more sound. Despite the difference, at least both coefficients are positive and within a reasonable range.

6.5 Conclusion

In this chapter, we have presented the results of the multi-start local search algorithm. We also provided an extensive analysis and interpretation of the results. This enables us to answer the following research subquestions:

5. *Does the more advanced profit-driven sort insertion heuristic contribute more to the quality of the solutions in the multi-start heuristic than the simple quantity sort insertion heuristic?*

6. *What local search operators contribute the most to the objective function?*

7. *What is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping?*

8. *Under what circumstances is the use of transhipment hubs effective?*

We found that the algorithm produces very different networks. There is a large spread in the profitability of the final networks, which is due to the multi-start characteristic of the algorithm. Some initial networks are constructed too weakly to end up as a good network, while others start off much better and become strong networks after local search has been applied.

In general, we see that revenues are rather high compared to the associated costs. That is mainly caused by the simplified cost structure we have chosen, which inevitably leaves out some cost variables. This also explains the rather large deviation between our best profit ($4.81 billion) compared to the expected Maersk 2010 profit ($1.095 billion).

Still, our results indicate that the multi-start local search algorithm is quite capable of producing sound networks. The network with the highest profit was produced using the 'Q & 1-2' configuration, which denotes the use of the Quantity Sort method, and local search operators 1 and 2 (route-length, and port-exchange operator respectively). This configuration is also the one that performs best on average, although the configurations 'P & 1-2' and 'P/Q & 1-2' perform similarly. That the algorithm is capable of producing sound networks also follows from our comparison with the upper bound of $5.09 billion, indicating a relative gap of 5.5% with the upper bound. Note that the optimal solution will have a lower profit than the upper bound, which means that the gap to the optimal solution will be smaller.

This brings us to answering the first subquestion:

5. *Does the more advanced profit-driven sort insertion heuristic contribute more to the quality of the solutions in the multi-start heuristic than the simple quantity sort insertion heuristic*?

The profit-driven sort method uses a profitability based scoring system to allocate demand. We expected that this method would yield better results. However, statistical analysis showed that there is no significant difference in network profits between the profit-driven sort method, and the much more simple quantity-based sort method. On the other hand, the network profits *before* the local search phase indicate that Quantity

Sort yields significantly better results. However, the difference is cancelled out by the local search phase. The difference in execution time between both sorting methods is negligible.

While the sort methods did not show different results with respect to profit, we witnessed completely the opposite with the local search operators. We are now able to answer the subquestion:

6. *What local search operators contribute the most to the objective function?*

The route-length operator shows an average improvement of 52% compared to the initial phase without local search, which indicates that this operator is very effective. When the operator is used on its own, or in conjunction with the port-exchange operator, its relative improvement is even higher than 75%. This success can be explained by its highly influential network adjustments and its profit-minded implementation. The port-exchange operator has the worst performance, it only improves performance by 1-2%. It appears that the limited remaining capacity on the routes is the bottleneck for this operator.

The transhipment operator on its own performs modestly. Its improvement of around 16-17% is not that bad. However, when combined with the route-length operator, the performance of the latter operator is heavily affected to the extend that their combined performance is worse than the application of the route-length operator alone. Statistical analysis confirms that the combined force of the route-length and port-exchange operator leads to the best results. The transhipment operator performs less than expected, since it incurs too much costs in the first phase of the operator, by rerouting demands, and cannot compensate for this extra cost in phase 2, when allocating new demand pairs.

When looking at execution time, the port-exchange operator is the most efficient, followed by the route-length operator. The transhipment operator is the least efficient, it needs much more time due to the increased number of path possibilities that arise when using transhipment.

The preceding conclusions also lead to the answer to the following subquestion:

7. *What is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping?*

We have seen that the use of the route-length operator in conjunction with the port-exchange operator leads to the best profits. We have also seen that the 'Q & 1-2' configuration yields the highest profit, both on average and based on the best network we found. However, statistical analysis showed that this configuration did not perform significantly better than the 'P & 1-2' and 'P/Q & 1-2' configurations. Therefore, we conclude that the configurations in which operators 1 and 2 are used, in that specific order, leads to the best results. The use of different sorting methods is not relevant. The Q & 2 configuration is the most efficient, since it needs the least computation time among all combinations, namely 20 seconds for one network. The 'Q & 1-2' configuration needs around 35 seconds to complete, which is still reasonable given its 68% higher effectiveness.

When we analyze the characteristics of profitable networks, we find that they share common properties. The most important factor is allocation. Profitable networks all have a high percentage of allocated containers. The number of routes in a network, and the route utilization are also relevant factors, although they affect profit less than allocation. The number of ports per route, and the fleet size have the lowest impact on profitability.

We also find a negative relation between the number of transhipped containers and the profit. This confirms the earlier conclusion that the transhipment operator is outperformed by other operators.

Finally, we will answer the last subquestion:

8. *Under what circumstances is the use of transhipment hubs effective?*

The definition of transhipment is transferring cargo from one ship to another, with the ships operating on different services, in order to continue the journey to its final destination. The transhipment takes place in a hub port, which is a port that is visited by both services. Introducing transhipment can improve the allocation of demand, and thereby the profit, in three ways. First, transhipment can reduce the path length from origin to destination. Second, transhipment can prevent unnecessary movements of cargo along the turning point. Third, transhipment enables unallocated demand to be allocated. These improvements result from the increase in path possibilities that comes with transhipment.

In general, the introduction of transhipment is effective as long as the revenue from the additional demands that can be transported covers the extra costs that are incurred. Anyway, in our benchmark study, the improvement of the transhipment operator was rather disappointing. We gave several explanations for this weak performance. It might be the case that the use of transhipment is not effective because there is not much demand left in the demand matrix to allocate. Another reason is that there is not enough space in the remaining capacities of the routes to successfully tranship this additional demand. Another reason is the implementation of the transhipment operator. We choose to first optimize the remaining capacities, such that the additional demand can better be allocated in the second step. However, since these steps are not combined, it may happen that too much demand is rerouted, while the space that is obtained is not used for new allocations. In this situation, the high rerouting costs cannot be covered. Costs might have been cut when combining the two steps, only rerouting demand when additional space is needed. Moreover, a cost-analysis should be part of the operator to make sure improvement takes place for each move.

We can summarize the circumstances that are necessary for an effective introduction of transhipment as follows. The additional demand that can be allocated with transhipment should be enough to cover for the costs. Furthermore, there should be enough remaining capacity on the ships in order to allocate the extra demand.

Chapter 7

Additional experiments

In this chapter we provide various additional experiments to the benchmark study from Chapter 6. These experiments will show the performance of the multi-start local search algorithm from a different angle, and justify some of the choices we have made. In Section 7.1, we provide a small benchmark of the multi-start local search algorithm against completely different algorithms, for which existing results are available. Since we are the first to use the Maersk Asia-Europe data set, it is difficult to place the results from Chapter 6 into perspective. However, using data from a previous study, we are able to compare the performance of the multi-start local search algorithm with other approaches. Section 7.2 compares the performance of different implementations for the transhipment operator. It provides two alternative implementations, and justifies the choice for the implementation that was used in the benchmark study. In the experiment, we will show how a single-hub model performs as opposed to our default multi-hub model. In Section 7.3, we provide the results of an additional experiment that uses a smaller ship size. In the benchmark study we used a fixed ship size of 12,000 TEU, but the rather low utilization rates on the routes made us wonder if a smaller ship size could increase the performance. We perform a small experiment using a ship size of 8,000 TEU, as well as an experiment with a mixed fleet, that give an answer to this question. Section 7.4 investigates the performance of our algorithm when varying initialization parameters for the multi-start phase. For example, it is interesting to know how the number of networks influences the final performance, such that a tradeoff can be made between effectiveness and execution time. In Section 7.5 we analyze the impact of re-applying local search operators. We provide multiple configurations to investigate whether it pays off to use the same operator more than one time. Finally, Section 7.6 concludes the chapter.

7.1 A seven-port model of the Asia-Europe trade lane

The Asia-Europe trade lane is a popular subject, given the numerous studies devoted to different logistical challenges on this trade lane. Its relevance to researchers, the huge volumes, and the trade imbalance make this trade lane an interesting subject. Although suitable data sets for liner shipping are difficult to obtain, as we have seen in Chapter 5, some smaller scaled examples exist.

In this experiment, we use a smaller data set and different parameters. Although these parameters are not anywhere close to reality, we use them to compare the results of the experiment with earlier results from another study.

In Man (2007), a self-constructed ten-port data set is introduced to test the performance of the heuristic that is proposed. This heuristic is based on the generic set covering problem, which is adjusted to account for the capacity constraint and the cargo allocation component involved with liner shipping. The implementation is completed and presented as the *KWM algorithm* in Lachner & Boskamp (2010). The latter study also proposes a new heuristic, the so-called *PDA algorithm*, which uses profit-driven criteria to allocate demand. For computational performance reasons, the ten-port data set was scaled down to a seven-port model and used to asses the performance of both algorithms.

In this section, we subject the seven-port data set to the multi-start local search algorithm as presented in Chapter 4, using the same parameter settings as defined in Lachner & Boskamp (2010). This allows us to compare the results. Before we present the results on the seven-port model, we will first elaborate on the data set, the model's assumptions, and provide the model's variable settings.

7.1.1 Assumptions and model variables

The seven-port data set originates from the ten-port data set as constructed by Man (2007). It represents the Asia-Europe trade lane, containing ports from Asia, the Middle East, and Europe. The distances between ports are based on the Sea Rates Distance Calculcator (2010). Demand between the ports is estimated by the original author, and not based on any real world data. The demand of the final seven-port data set is shown in Table 7.1.

The majority of the assumptions we provided in Chapter 3 are also applied in Lachner & Boskamp (2010). However, we have to identify the differences and their possible impact, because the outcome is likely to be biased due to different assumptions.

First of all, revenue is defined differently. The revenue per TEU is defined as a sum

Table 7.1: Demand matrix of the seven-port model (in 1,000 yearly TEU's)

O/D	Tokyo	Shanghai	Singapore	Jebel Ali	Antwerp	Rotterdam	Hamburg
Tokyo	0	0	87	23	100	223	138
Shanghai	0	0	92	18	151	631	364
Singapore	118	131	0	24	149	358	277
Jebel Ali	42	28	21	0	46	51	39
Antwerp	102	132	72	0	0	0	0
Rotterdam	110	501	155	8	0	0	0
Hamburg	98	280	123	3	0	0	0

per *traveled* mile, as opposed to our assumption that revenue is calculated based on the *direct* distance between the origin and destination of the cargo. It is the multi-start local search algorithm that will suffer from this inconsistency, because the direct distance can never be higher than the traveled distance, while the other way around is often the case. Therefore, we will by average experience lower revenues.

Second, we assume that because of the trade imbalance, revenue is asymmetric, which means that revenue depends on the direction cargo travels. However, we can overcome this difference by adjusting our revenue variables such that the two directions yield equal, symmetric, revenues.

Variable settings can easily be modified, unlike the assumptions that are the foundation of an algorithm. The cost structures of the PDA and KWM algorithm match ours. However, the variable settings are quite different. We will have to adjust our settings to avoid any inconsistencies. For example, having a higher or lower revenue per TEU strongly affects the outcome of an algorithm, in such a way that algorithm outcomes cannot be compared anymore. In Table 7.2 we provide the original variable settings from Chapter 6, along with the adjusted settings that are used in this experiment.

7.1.2 Results

In this section we discuss the results from both the KWM and PDA algorithm, as well as the results from the multi-start local search algorithm.

The report of Lachner & Boskamp (2010) provides a case example, which shows results for one of the nearly 700 different configurations of their benchmark study. This case example conveniently shows the output variables so that we can compare results from all the three algorithms in detail. It should be mentioned that the implementation of the KWM and PDA Algorithm from Lachner & Boskamp (2010) contained an inaccuracy in the profit calculation part, which not only results in low profits, but also

Table 7.2: Adjusted model variable settings

	Original settings	Adjusted settings
Sailing speed	20 knots	26 knots
Ship capacity	12,000 TEU	10,000 TEU
Capital costs	\$11,350,000/yr	\$18,000,000/yr
Operational costs	\$2,190,000/yr	0
Fuel costs	\$150/nm	\$100/nm
Port costs	\$ 25,000/call	\$200,000
Handling costs	\$175/TEU	0
Revenue (westboud)	\$0.1677/nm/TEU	\$1.0417/nm/TEU
Revenue (eastbound)	\$0.0838/nm/TEU	\$1.0417/nm/TEU
Minimum slack time	1 day	2 days
Port delay	12 hours	20 hours

causes unbalanced performance among the two algorithms. We have fixed the error and recalculated the results for the single case example. We will use these results below instead of the original results.

The multi-start local search algorithm uses two sorting methods (Quantity Sort and PDA Sort), and three local search operators. However, one of the local search operators, the *transhipment* operator, contains elements that are not covered in the KWM and PDA algorithms. These algorithms do not support transhipment, so it would be unfair to use the transhipment operator. Hence, we have three[1] sorting methods and two operators, of which we include the different sequences, resulting in 12 different configurations.

In Table 7.3 we show the results from all three algorithms. The best configuration for the multi-start local search algorithm in this case turns out to be 'P/Q & 2'. which means: randomly use PDA Sort or Quantity Sort, and use the port-exchange local search operator.

The results indicate that the profit increases with more than 35% when we use the multi-start local search algorithm, compared to the KWM and PDA algorithms. That is mainly because of the increased ship utilization: with roughly the same fleet, we transport 94.9% of the demand, as opposed to only 63.9% in case of the PDA algorithm.

From Table 7.3 it also appears that the execution time increases dramatically with the new algorithm. However, the execution time of the PDA algorithm rises exponentially with the number of ports involved, due to the calculation of all possible route

[1]We also consider a method that chooses randomly between Quantity Sort (Q) and PDA Sort (P) for each network, denoted by P/Q.

Table 7.3: Algorithms' results of the 7-port model

	KWM	PDA	Multi-start local search
Execution time (sec)	1.5	97.2	18,318
Profit	\$66.7 billion	\$67.8 billion	\$92.7 billion
Demand fulfilled (TEU)	7,003,300	6,008,000	8,913,393
Demand fulfilled (%)	74.6%	63.9%	94.9%
Number of routes	16	9	10
Fleet size	81	60	66
Avg. ports per route	4	7.2	5.1
Distance traveled (nm)	282,177	197,883	229,785
Number of paths	51	15	60
Total revenue	\$68.2 billion	\$68.9 billion	\$94.6 billion
Total costs	\$1.49 billion	\$1.1 billion	\$1.9 billion
Best configuration	-	-	P/Q & 2
Avg. utilization per route	36%	71%	75%
Fully used legs (total)	5 (36)	27 (56)	23 (51)
OD pairs covered (total)	33 (33)	13 (33)	33 (33)

combinations. The number of possibilities is referred to as the permutation problem, and is equal to $\sum_{k=1}^{n} \binom{n}{k}$, where n denotes the number of ports in the data set. The PDA algorithm allows problems up to 7 or 8 ports, before the problem becomes computationally too expensive to solve. The multi-start local search algorithm doesn't suffer from this problem, so that real life examples with over 60 ports can be easily solved within a reasonable amount of time. Moreover, we have simulated all twelve different configurations, one for each possible sorting method and local search operator sequence. Once the best configuration is known, one may use that one for future simulations, so that the execution time is reduced to roughly $\frac{1}{9}$th. In other words, although the execution time of the example above is relatively high, the multi-start local search algorithm is very scalable.

Another striking figure is the low costs compared with the total revenue. Each algorithm shows that total costs are only about 2% of the total revenue. Obviously, this proportion is not very realistic. The main reason for this outcome is due to the high revenue we have assumed: over \$1/nm/TEU. This is almost ten-fold the revenues we find in real-life data. However, to compare results of the multi-start local search algorithm with the KWM and PDA algorithm, we have set the cost and revenue variables equal to the settings used in previous studies. This leads to abnormal cost-revenue proportions.

Still, the conclusion holds that the multi-start local search algorithm performs over 35% better than the other two algorithms.

Table 7.4 shows the results of all configurations, as well as the performance of each local search operator. These results show that operator 2 performs better than operator 1. The difference with the benchmark study in Chapter 6, where we obtained over 50% profit increase with operator 1 and only 1-2% with operator 2, is striking.

Table 7.4: Multi-start local search algorithm: results of all configurations

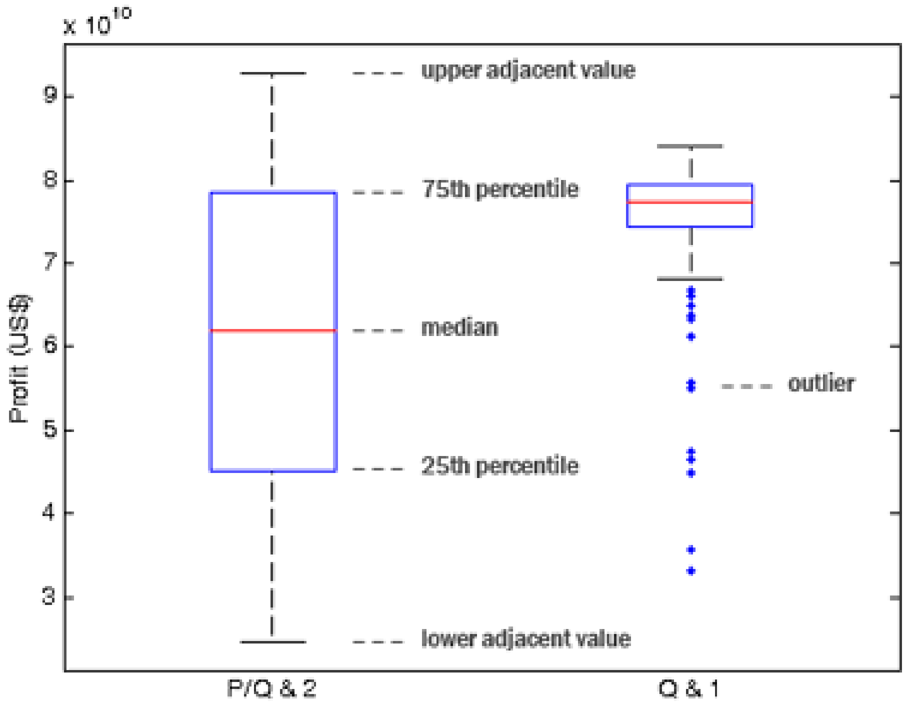
Configuration	Profit (best)	Profit average	Increase operator 1	Increase operator 2
P & 1	\$82.8 bn	\$69.3 bn	2.2%	-
P & 2	\$90.8 bn	\$66.2 bn	-	7.0%
P & 1-2	\$89.7 bn	\$68.6 bn	2.6%	4.5%
P & 2-1	\$91.9 bn	\$66.5 bn	3.6%	6.5%
Q & 1	\$84.1 bn	\$74.3 bn	2.3%	-
Q & 2	\$86.1 bn	\$58.6 bn	-	3.7%
Q & 1-2	\$89.9 bn	\$59.0 bn	6.5%	1.7%
Q & 2-1	\$88.4 bn	\$60.3 bn	4.0%	3.1%
P/Q & 1	\$81.7 bn	\$70.6 bn	2.5%	-
P/Q & 2	**\$92.7 bn**	**\$61.2 bn**	**-**	**4.8%**
P/Q & 1-2	\$89.4 bn	\$68.5 bn	3.9%	3.6%
P/Q & 2-1	\$91.1 bn	\$64.8 bn	2.7%	5.0%

In this experiment, we used different cost and revenue parameters. Obviously, this affects the behaviour of the algorithm. The algorithm, and in particular the operators, are sensitive to different parameters. One of the reasons operator 1 (the route-length operator) performs so much worse, can also be found in the parameter settings of the model. Recall that this operator tries to remove cost-inefficient ports and to add new ports to allocate additional demand. When we take a look at the cost and revenue parameters, we see that the revenues are much higher than the original (realistic) settings. On the other hand, port costs are considerably higher, but that effect is more or less neutralized by the handling costs which are set to zero. In other words, the revenues are too high compared to the cost parameters.

When we think of the route-length operator, we expect that it will hardly remove any ports, because the revenues will most often exceed the costs. Hence, no profit increase is obtained by removing ports. On the other hand, we would expect that new ports are added, because it seems quite easy to make money from additional demand, due to the high revenue rate. But even with the very modest performance of both operators, we see that the average percentage of fulfilled demand is almost 95%. That leads to the conclusion that initial networks, before the local search phase, already have most of the demand allocated. In other words, it will be difficult for the route-length operator to find demand that it can allocate by extending a route with an additional port, especially because it also relies on the remaining capacities on those routes.

We observe that the port-exchange operator, which only moves ports, in this situation yields a better performance.

Figure 7.1: Profits of P/Q & 2 and Q & 1 configurations compared



The results also show that the winning configuration, P/Q & 2, has a rather low average profit. This contradicts the results from the benchmark study, where the configuration with the highest average profit also produced the best network. One could wonder whether the best network, with a profit of $92.7 billion, is some sort of 'outlier', or a coincidence. To answer this question, we take a closer look at the profit distribution of the configurations. Figure 7.1 shows the boxplot of both the best configuration (P/Q

& 2), and the configuration with the highest average profit (Q & 1). These boxplots show the median value as well as the 25th and 75th percentiles, also known as lower quantile or lower hinge, and upper quantile or upper hinge, respectively. The lower and upper *adjacent values* depict the smallest and largest values in the data set, that are not outliers. Outliers are defined as values that are beyond 1.5 times (upper quantile - lower quantile). Hence, in the left boxplot, where no outliers are found, the adjacent values represent the extreme values. On the contrary, the right boxplot contains a lot of outliers below the lower quantile, so that the lower adjacent value is not equal to the smallest value in the data set.

The figure shows that the profit distribution of the Q & 1 configuration is much more narrow compared to the P/Q & 2 configuration. Also, the latter configuration has more extreme values. We observed that over 15% of the 100 networks has a higher profit than the best network in the Q & 1 configuration. Hence, the best network cannot be considered a coincidence. Although many other configurations have higher averages, the P/Q & 2 configuration produces some high quality solutions, of which one is the best network overall.

7.2 Different transhipment operator implementations

In Chapter 6 we discussed how to model transhipment in routing and scheduling of liner shipping operations. Following from that discussion, we distinguish three main approaches to model transhipment. In this experiment, we will present the other two approaches, and compare the performance of these three different implementations of the transhipment operator. Furthermore, we decompose the performance of the operators into parts in order to understand their effect on the total result. This allows us to interpret the success of the chosen implementations, and propose additional modeling concepts. First, we will discuss the differences of the three implementations. Then, we will present the experimental design. Finally, we report on the results of the experiments and provide an analysis of the performance.

7.2.1 Transhipment operator implementations

The three approaches differ with respect to the number of ports fulfilling the hub function, and the number of transhipment moves allowed. We distinguish the single-hub single-move variant, the multi-hub single-move variant, and the multi-hub multi-move variant. The approaches should deliver different performances, where the underlying idea is that a better solution can be obtained at a higher computational cost.

All approaches consist of the same two steps, only the implementation differs. In the first step, allocated demand pairs that satisfy certain criteria are rerouted via transhipment hubs. In the second step, demand that could not be allocated before is now allocated using transhipment. For the details and rationales behind these steps we refer to Section 4.3.4. Next, we discuss the specifics of each approach.

Single-Hub Single-Move Transhipment

With *single-hub single-move* transhipment, we consider only one transhipment hub, and only one transhipment move is allowed. Since only one port in the network fulfils the hub function, we need to determine the best port for this task. In this implementation we have chosen to assign the hub status based on the demand matrix. Using the demand matrix, we calculate each port's throughput, and the port with the highest throughput becomes the hub. Then, we run the two steps described above, and only search for transhipment possibilities via the chosen hub, for a single transhipment move. This means the transhipment path consists out of the origin-hub path and the hub-destination path. Since we consider only a singe hub and allow a single transhipment move, the execution time will be the lowest of the three approaches. Furthermore, less transhipment path possibilities are available, such that we expect this approach to underperform against the other two approaches.

Multi-Hub Single-Move Transhipment

With *multi-hub single-move* transhipment, all ports are transhipment hubs, but we only allow one transhipment move. Assigning all ports the transhipment status may seem too much, but remember that not all ports have to be used as hub. Having multiple hub ports is a big advantage over the use of a single hub, since the number of transhipment possibilities increases considerable. What changes in comparison with the single-hub single-move variant is that in this case we search for transhipment possibilities via each (hub) port on the interval [origin,destination]. Again the transhipment path consists out of the origin-hub path and the hub-destination path. We already have seen the implementation of this approach in Chapter 4, and this is the one used in the benchmark study in Chapter 6. The execution time of this approach is in between the execution time of the single-hub single-move and multi-hub multi-move approach.

Multi-Hub Multi-Move Transhipment

With *multi-hub multi-move* transhipment, all ports are transhipment hubs, and we allow multiple transhipment moves. This is the most complex, but also the most realistic concept. Instead of obtaining a transhipment path consisting of the origin-hub path and the hub-destination path, like with the previous approaches, we can generate transhipment paths of any length. For example, we might obtain a transhipment path consisting of the paths origin-hub1, hub1-hub2, hub2-hub3, and hub3-destination. Allowing multiple transhipment moves increases the number of transhipment possibilities greatly, such that we expect to find the best solutions using this approach. However, the disadvantage is that the computational cost needed to enumerate all these possibilities is much higher. In order to prevent the execution time to become too large, we limit the number of transhipment moves to 5. Since the operator prefers the shortest transhipment paths, and we expect most demand pairs to be connected within 5 transhipments, this will have almost no influence on the final result.

The precise working of the multi-hub multi-move operator is as follows. It follows the same steps as the other variants, only the implementation differs. This difference lies in the way we search for transhipment possibilities. In the case of multi-move transhipment, we generate all possible transhipment paths between the origin and destination. We iterate through all ports on the interval [origin,destination] and keep a list of transhipment path candidates.

For example, suppose we have to find multi-move transhipment paths for demand pair 3-7. Then, we start with port 3 and store the routes that contain this port. Suppose port 3 occurs on route A and C, then our path candidates list contains 3(A) and 3(C). Next, we do the same for port 4, but now we can only extend existing path candidates. Suppose port 4 occurs on route A and B, then our path candidates become 3(C), 3(A)-4(A), and 3(A)-4(A)-4(B). Note that we can tranship in port 4 from route A to B, however we can also remain on route A, therefore we end up with these candidates. Suppose port 5 occurs on route B and C, then our path candidates become 3(C)-5(C), 3(A)-4(A), 3(A)-4(A)-4(B)-5(B). Note that 3(A)-4(A) is unchanged and remains in the list for future ports. We continue this process until we reach the destination port. Then, we finalize the path candidates that are able to end at the destination port, and throw away all others, obtaining the list with all correct transhipment possibilities. Then, we throw away candidates that do not have enough remaining capacity, and sort the candidates by length, i.e., the number of ports visited. Recall that we set a limit of 5 transhipment moves, in order to lower execution time. As a result, some path candidates may become invalid during the process and are deleted.

7.2.2 Experimental design

In this experiment the objective is to gain insight into the performance of the different transhipment operator implementations. With performance we mean effectiveness (profit), but we are also interested in the efficiency (execution time). In the benchmark study from Chapter 6 we used the multi-hub single-move transhipment operator, since we wanted to obtain a good solution in a reasonable amount of time. With this experiment, we can find out whether or not it was the right choice to use that operator implementation.

The experiment consists of a number of simulations in which we create an initial service network, on which we run all three transhipment operator variants separately. The initial solutions are generated using the first phase of the multi-start local search heuristic that is described in Chapter 4. In order to obtain unbiased results we choose the 'P / Q' parameter, indicating a randomly chosen insertion heuristic, i.e., either PDA sort or Quantity sort. Once the initial solution is obtained, it will be used as starting point for all three operator variants.

We run 10 of these simulations[2], after which we compute the average results. In addition, we perform two extra runs of 10 simulations, in which either the first or the second step of the transhipment operator is left out. Recall that the first step focused on rerouting existing demand, whereas the second step focused on allocating remaining demand from the Demand matrix. This will give us more insight into the influence of these steps on the performance.

For this experiment we use the same model variables and algorithmic parameters as presented in Table 6.1, except for the number of service networks, which is set to 10. We also use the MAE data set, consisting of 58 ports from the Maersk Asia-Europe network during spring 2010, of which a more detailed specification is available in Chapter 5.

Following the characteristics of the three implementations, we expect the single-hub single move variant to perform the worst, and the multi-hub multi-move variant to perform the best. The performance of the multi-hub single-move variant will be in between. Furthermore, we expect the multi-hub multi-move variant to be computationally most intensive, i.e., least efficient, and the single-hub single-move variant to be the most efficient. We expect the efficiency of the multi-hub single-move variant to be in between.

[2]The number of 10 is chosen in order to prevent the total running time to become too long. We acknowledge this is not enough to obtain statistically reliable results, but at least it gives an indication of the performance variation among the implementations.

7.2.3 Results

In this section we discuss the results of the experiment, where we compare the performance of three different implementations of the transhipment operator. Table 7.5 shows the results of the main experiment, as well as the experiments in which only one step is activated.

Table 7.5: Results of the transhipment operator implementations

	Initial Network	SHSM	MHSM	MHMM
Experiment with both steps				
Avg. Profit increase	-	**8.8%**	**16.7%**	**7.3%**
Min. Profit increase	-	0.0%	1.3%	-10.3%
Max. Profit increase	-	14.2%	30.9%	29.5%
Avg. Profit	\$2.639b	\$2.867b	\$3.067b	\$2.805b
Avg. Revenue	\$5.294b	\$5.712b	\$6.315b	\$6.542b
Avg. Cost	\$2.654b	\$2.845b	\$3.248b	\$3.737b
Avg. Demand	68.0%	72.7%	81.0%	83.5%
Avg. Demand transhipped	-	14.5%	44.9%	48.1%
Avg. Paths transhipment	-	492.1	1483.2	2208.2
Avg. Exec. time (sec)	-	6.36	107.94	2254.73
Min. Exec. time (sec)	-	2.28	61.64	85.63
Max. Exec. time (sec)	-	12.06	174.65	16635.73
Experiment without step 2 (Allocate remaining demand)				
Avg. Profit increase	-	-2.84%	-12.71%	-25.57%
Avg. Profit	\$2.494b	\$2.419b	\$2.174b	\$1.848b
Avg. Revenue	\$5.418b	\$5.418b	\$5.418b	\$5.418b
Avg. Cost	\$2.924b	\$2.967b	\$3.244b	\$3.570b
Avg. Demand	69.53%	69.53%	69.53%	69.53%
Avg. Demand transhipped	-	5.30%	37.43%	39.63%
Avg. Exec. time (sec)	-	6.43	78.05	2713.37
Experiment without step 1 (Reroute existing demand)				
Avg. Profit increase	-	14.52%	24.86%	23.26%
Avg. Profit	\$2.845b	\$3.239b	\$3.544b	\$3.496b
Avg. Revenue	\$5.916b	\$6.456b	\$6.901b	\$6.976b
Avg. Cost	\$3.071b	\$3.217b	\$3.356b	\$3.481b
Avg. Demand	75.74%	81.89%	87.77%	88.72%
Avg. Demand transhipped	-	7.50%	13.87%	14.85%
Avg. Exec. time (sec)	-	5.73	52.27	1258.37

When considering the main experiment, we notice that the multi-mub single-move (MHSM) transhipment operator, that we used in Chapter 6, turns out to be the most effective. It actually performs much better, on average, than the single-hub single-move (SHSM) and multi-hub multi-move (MHMM) operators. Moreover, it is the only operator of which the profit increase kept positive over all 10 runs, whereas with the other two operators we observe no profit increase or a negative increase. The reasons for this negative performance become more clear later, when looking at the other two experiments. The MHSM operator has the highest profit of all operators, followed by the SHSM operator. The MHMM operator, which we expected to perform best, actually has the least profit, due to the high amount of handling costs that it incurs. It does yield the most revenue, but it turns out it cannot keep the costs down, resulting in a overall result that is worse than it is for the other operators.

The average demand percentage that is fulfilled corresponds to the operator characteristics. The SHSM operator can fulfil less demand than the MHSM operator, since it can create less transhipment possibilities. The same goes for the MHSM operator, which can fulfil less demand than the MHMM operator. This is also visible from the average number of transhipment paths.

When looking at the efficiency, the SHSM operator has the lowest execution time and the MHMM operator has the highest, just as expected. The efficiency of the MHSM operator is in between, but much closer to the SHSM operator. The distribution of the execution time for the MHMM operator is very wide. The maximum running time for the MHMM operator over the 10 runs was 16635 seconds, which comes down to 4.6 hours. This indicates the problem with using this operator in a multi-start approach. When creating 100 networks, the MHMM operator may cause the total running time of the multi-start local search algorithm to take weeks.

The results of the other two experiments showed in Table 7.5 give us some insight in the performance of the steps that each operator consists of. First, we ran an experiment where step 2, in which we allocate remaining demand, was left out. From this result we can see that, as we expected in Section 6.3, a negative profit increase is obtained in step 1. This is due to the fact that step 1 only reallocates demand that was already allocated. The goal of the reallocation process is to gain more space on the ships, in order to allocate more demand in step 2. As a result, we gain no revenue, but do incur additional cost, and the MHMM operator does so more than the other two operators. Second, we ran an experiment where step 1 was left out. Remarkable is the profit increase for the MHSM operator, that is slightly higher than the MHMM operator. When looking at the revenues and cost, we can see that the MHMM operator does yield the most revenue, but the additional handling cost incurred are too high to obtain the highest profit.

At this point we are able to judge our decision to use the multi-hub single-move transhipment operator in Chapters 4 and 6. Based on this experiment, the MHSM operator turns out to be the most effective, and reasonably efficient. We did not expect the MHSM operator to perform better than the MHMM operator when it comes to effectiveness, but the additional costs incurred from the extra handling activities were such high that it prevented a good overall result. Furthermore, this experiment clearly shows that too much profit is lost in step 1 of the transhipment operator. Prior to this experiment, it seemed reasonable to first gain space in the ships, by rerouting demand that unnecessarily visits the turning point and could be allocated to shorter paths via transhipment. Then, step 2 would be better able to allocate demand that could not be allocated to direct paths. However, after running this experiment, we have to conclude that maybe too much demand is rerouted. It might have been better to only reroute demand where additional space was needed by combining steps 1 and 2. This would keep the extra handling cost incurred to a minimum. Moreover, a profit-loss calculation should be incorporated in the transhipment operator in order to prevent profit decreasing moves. Focusing on a higher level, it might be better to let the transhipment operator adjust the network, like the route-length and port-exchange operators do. At the moment, the transhipment operator only focuses on reallocating demand, whereas changing the network based on certain demand characteristics may yield better transhipment possibilities. We propose these improvements for future research.

7.3 Using smaller sized ships

In Chapter 6 we have seen that, using 12,000 TEU ships, the best network has an average utilization rate of 59.8%. The average utilization rate for the Q & 1-2 configuration overall is worse, only 45.4%. In addition, Section 6.4.2 points out that there exists only a modest positive relationship between the utilization rate and a network's profit.

We know from reality that the utilization rate is an important issue. We did not incorporate the problem of *empty container repositioning* into our model, but it is often used to boost utilization rate on one hand, and deal with trade imbalances on the other hand. Still, the fleet that Maersk uses on the Asia-Europe trade lane is composed out of smaller and larger sized ships, with an average capacity of almost 8,500 TEU. Although the actual utilization rate is unknown, this leads to the question whether it would be more profitable to use smaller sized ships.

In the benchmark study from the previous chapter we assumed a fixed ship capacity of 12,000 TEU. This choice follows the trend of larger container ships on the one hand, and avoids the computational intensity of mixed ship capacities on the other hand. In this experiment we examine the impact of a smaller ship size on the profitability.

7.3.1 Using a fixed ship capacity of 8,000 TEU

Using a mixed fleet (i.e., using a fleet with both smaller and larger ships) provides the flexibility to match the ship size to the demand on a certain route. However, solving the allocation problem for different ship sizes is computationally very intensive.

In this part of the experiment we change the ship size from 12,000 TEU to 8,000 TEU. Using a fleet with fixed, smaller sized ships, we examine the impact on the profitability for the Q & 1-2 configuration. We are primarily interested in the resulting profit together with the utilization rate. In other words, will the use of smaller ships result in higher utilization, and if so, will this be more profitable?

To examine the impact of the use of smaller sized ships, we use the Q & 1-2 configuration. We maintain exactly the same variable and parameter settings as defined in Chapter 6, except for the ship size. Instead of 12,000 TEU we limit the ship capacity to 8,000 TEU. Then, we build 100 networks using the multi-start local search algorithm. We compare both the average performance as well as the performance of the best network, and analyze their profitability, utilization rate, and network composition, such as number of routes and the string length.

We use the following adjusted parameters. The yearly capital costs for a 8,000 TEU vessel are equal to \$8.35 million. A smaller vessel also consumes less fuel, such that we assume the fuel costs are reduced to \$120/nm. For a more detailed explanation of these parameter settings, we refer to the next subsection. Smaller ships are less cost-efficient, which shows from the costs per container per mile. When we consider the best network of the benchmark study, the details provided in Table 6.7 show that a fleet of 69 vessels together sail 222,842 nm per week. Hence, on average one ship sails 167,939 nm per year. The yearly costs of a 12,000 TEU ship are \$11.35m, which corresponds to $\$11,350,000/167,939) = \$67.58$ per nm. When fuel costs (\$150) are added, the costs are equal to \$217.58 per nm. Similarly, if we compute the costs for a 8,000 TEU vessel, we obtain $(\$8,350,000/167,939) + \$120 = \$169.72$ per nm. When we divide these figures with the ship capacities, we obtain \$0.018 and \$0.021 for the 12,000 TEU and 8,000 TEU ship respectively. This confirms that the costs per TEU per mile are higher when smaller ships are used. In order to compensate for this, a higher utilization rate has to be obtained.

In Table 7.6 we present the results. To allow for easy comparison, we provide both the new 8,000 TEU results, and the 12,000 TEU results from Chapter 6. Furthermore, we provide both the average figures and the figures corresponding to the network with the best profit.

Table 7.6: Results of the Q & 1-2 configuration using 8,000 TEU ships

	8,000 TEU	12,000 TEU
Profit (average)	\$4.00 billion	\$4.24 billion
Profit (best)	\$4.54 billion	\$4.81 billion
Utilization (average)	72.4%	45.5%
Utilization (best)	69.8%	59.8%
Fleet size (average)	89.2	85.6
Fleet size (best)	103	69
Number of routes (average)	10.5	10.0
Number of routes (best)	12	8
Avg. ports per routes (average)	25.4	26.0
Avg. ports per routes (best)	26	27.6
% unallocated containers (average)	12.6%	5.5%
% unallocated containers (best)	1.8%	3.0%
Total revenue (average)	\$6.72 billion	\$7.34 billion
Total revenue (best)	\$7.65 billion	\$7.57 billion
Total costs (average)	\$2.72 billion	\$3.10 billion
Total costs (best)	\$3.11 billion	\$2.76 billion

The results show that the ship utilization is improved considerably when we use the smaller 8,000 TEU ships, both on average as well as in the best network. At the same time, we can see that the fleet has expanded, especially in the best network. There are 103 ships used in the 8,000 TEU case as opposed to the 69 ships in the 12,000 TEU case, which leads to increased capital costs. Total costs have risen over 12%. Revenues increased slightly to \$7.65 billion, because more containers could be allocated. However, the rise in revenues cannot compensate for the additional costs. In the best network, profit has dropped with over 6% to \$4.54 billion.

The network composition is more or less the same. The networks consists of slightly more routes, which are (on average) a bit shorter, but these differences appear to be rather insignificant.

In summary, the use of smaller ships leads to an expansion of the fleet. The associated rise in costs is not completely covered by the improved utilization rate and the increased number of allocated containers. This results in a deterioration of the profit with more than 6%. We conclude that a fixed ship size of 8,000 TEU performs worse than its 12,000 TEU alternative. Of course, in reality, fleets are usually composed of mixed ship sizes. For example, the Maersk fleet used in the MAE data set consists of ships with a capacity between 6,251 TEU and 14,770 TEU. In the second part of this experiment, we try to optimize individual routes by introducing a mixed fleet.

7.3.2 Optimizing individual routes using a mixed fleet

In this second part of the experiment we apply a more advanced technique. Instead of using a fixed ship capacity, we analyze each route of a network and determine the optimal ship size. Hence, we obtain a (heterogeneous) fleet with mixed ship capacities. This works as follows. Each route has a utilization rate between two ports, which is based on the use of a 12,000 TEU ship. When the maximum utilization rate on the route is below 100% we can use a smaller ship, because clearly, the ship is never completely full. In that case, we lower the ship's capacity in steps of 1,000 TEU, until it cannot be lowered any more without exceeding the 100% utilization constraint. However, this is just the first step, without any difficult computations involved. For example, when we analyze each of the eight routes in the best network of the Q & 1-2 configuration, we observe that only one route has a maximum capacity such, that a 8,000 TEU ship can be used. For the other seven routes we need capacities of more than 11,000 TEU so that we cannot lower the capacity on those routes. Therefore, we take our technique a level further.

The second step is to calculate the trade-off between a smaller capacity and the loss of cargoes. In other words, we compare the savings that are obtained using smaller ships, with the loss in revenues when we remove some of the cargoes from a route. This seems sensible, because utilization rates tend to be very volatile and are high during a small part of the cycle (e.g., when leaving Asia). The sample route in Figure 6.3 is such an example. We see that utilization rates are rather low on the major part of the route, but all of a sudden, between Hong Kong and Jebel Ali the ships are used more than 90%. When we remove some of the allocated cargo such that the maximum capacity never exceeds (for example) 75%, we can replace the 12,000 TEU ships that are currently being used on that route by smaller and cheaper 9,000 TEU ships. However, removing cargo from a route is costly because we lose revenue. Whether it is also beneficial depends on the amount of cargo we have to remove and how profitable that cargo was. Sometimes, we can reallocate (part of) the cargo to other routes. In that case, we reduce the loss in revenues.

Of course we have to use different cost parameters for each ship size. There are three cost parameters that vary with the ship size: capital costs, operational costs, and fuel costs. We chose to keep the operational costs unchanged, because it is difficult to set the correct values for that parameter. Ships in the 8,000-12,000 TEU region tend to have comparable operational costs, because they have similar crew sizes. Ships that are smaller than 8,000 TEU tend to have less costs, but we rather avoid that discussion and use the same operational costs. Obviously, the capital costs are different as well. Using equations 6.1 and 6.2, we have calculated the yearly capital costs for each ship size. Similarly, we calculate the fuel costs for different ship sizes. Fuel consumption is linearly related with the power of a ship's engine. According to a report of MAN B& W Diesel A/S (2008), the specified engine power of a 6,000 TEU ship is 53,800 kW, while it is approximately 60% more in case of a 12,000 TEU ship. Although there is not a perfect linear relationship between ship capacity and engine power, for the sake of simplicity we assume there is, so that we can easily calculate the fuel costs for any ship size.

Engine power grows with 60% when ship capacity increases from 6,000 to 12,000 TEU. Similarly, fuel costs have to rise 60% as well. We assumed the fuel costs to be \$150/nm when ship capacity is 12,000 TEU. We can calculate the savings in fuel costs for every 1,000 TEU capacity reduction as follows: $\$150 \cdot (60/160) \cdot (1/6) \approx \$9.50$. Hence, we have to subtract roughly \$9.50/nm for every 1,000 TEU of capacity reduction. Table 7.7 shows the yearly capital costs, as well as the fuel consumption we used for the differ-

Table 7.7: Using mixed ship sizes: costs per capacity

Capacity (TEU)	Yearly capital costs	Fuel consumption	$/nm/1,000 TEU
12,000	$11.35m	$150.00/nm	$18.13
11,000	$10.6m	$141.50/nm	$18.51
10,000	$9.87m	$131.00/nm	$18.98
9,000	$9.12m	$121.50/nm	$19.53
8,000	$8.35m	$112.00/nm	$20.21
7,000	$7.55m	$102.50/nm	$21.07
6,000	$6.73m	$93.00/nm	$22.18
5,000	$5.87m	$83.50/nm	$23.69
4,000	$4.97m	$74.00/nm	$25.90
3,000	$4.00m	$64.50/nm	$29.44
2,000	$2.95m	$55.00/nm	$36.28
1,000	$1.76m	$45.50/nm	$55.98

ent ship sizes. In the third column, we provide the costs per 1,000 TEU per nautical mile.

The technical approach is as follows. Using the steps explained above, we analyze each route in all 100 networks of the Q & 1-2 configuration. First, we try to reduce the ship capacity without removing cargo. The resulting profit, which can never be lower than the original profit with 12,000 TEU capacity, is set as a benchmark.

Next, we find the leg of the cycle where the utilization rate is the highest. The objective is to reduce that rate, so we obtain the target level of this utilization rate first. This target is initially equal to the utilization rate of the second-busiest leg, but we optimize that rate with respect to the full capacity. For example, suppose that we need a capacity of 11,400 TEU to cope with the busiest leg, and that we need 10,700 TEU for the second-busiest leg. Since we reduce ship capacity in steps of 1,000 TEU, we will set the capacity at 11,000 TEU, which is feasible because we remove at least 400 TEU from the busiest leg.

Once the target is set, we remove cargo to lower the utilization rate of the busiest leg. We do this by finding all cargo that passes the leg with the highest utilization rate, and we sort them according to profitability (i.e., according to distance between origin and destination). Then, we remove cargo until we reach the target utilization.

In the next step, we try to reallocate the cargo that we just removed to other routes of the same network. Obviously, unallocated cargo means lost revenue, so reallocation is much more preferred. Iterating through all other routes, we try to reallocate the cargo.

After removing (and sometimes reallocating) cargo, we obtain a reduced ship capacity. We compare the profit of the updated route with the initial route. If the new profit is higher, we save the modifications and keep the new ship capacity. If not, we do not save them, but we still proceed with the next step.

So far, we reduced the utilization of the busiest leg, which is the part of the route with the highest utilization rate. If the reduction was profitable, we saved the changes. In the next step, we are going to reduce the two busiest legs. Similarly to the previous steps, we first set the utilization *target*. We use the utilization of the third-busiest leg as a basis of this target, and then optimize it just like before. We then remove (and try to reallocate) cargo such that the utilization of the first and second-busiest leg is reduced to match the target. We repeat these steps, each time adding one leg, until we reduced all legs to the level of the one with the lowest utilization rate.

The previous steps form a iterative process. Sometimes, it is not profitable to reduce capacity, so that the route stays unchanged. Other times we successfully reduce the capacity, either at an early stage, or after multiple iterations.

In Table 7.8 we provide the results of the experiment with mixed ship sizes. We clearly observe that a substantial gain in profit is obtained. A new best network is found, with a profit of \$4.97 billion. On average, the profit increase is 8.7%. We also see that the percentage of unallocated containers has slightly increased, indicating that around 23,000 TEU of each network cannot be reallocated after removal.

Not surprisingly, the utilization rate increased, from 45.5% to 59.6%. We find that the average ship capacity across the 100 networks is now around around 8,700 TEU, instead of 12,000 TEU. This more or less corresponds to the mixed fleet of Maersk, which has an average capacity of almost 8,500 TEU. In the best network, we now have an average ship capacity of 7,154 TEU. Out of the eight routes, only one still has a capacity of 12,000 TEU. The other capacities vary between 3,000 and 11,000 TEU.

In the first part of this experiment, we tried to use smaller, but fixed ship sizes. This turned out to worsen the performance. In this second part, we used a fairly simple technique to reduce ship capacities based on lowering maximum capacity. Although a small part of the cargoes was removed, this approach worked. The average ship capacity dropped dramatically, from 12,000 TEU to approximately 8,700 TEU. This lead to savings in both the capital costs and the fuel costs. On average, we gained 8.7% in profit.

The best network gained a staggering 22.5% profit increase to \$4.97 billion. This is 3.4% higher than the performance of the former best network from Chapter 6, which had a profit of \$4.81 billion. The utilization rate of 59.2% is acceptable, given the fact that

Table 7.8: Result of using mixed ship sizes

Statistic	Result
Best network	
Profit	\$4.97 billion
Profit increase	22.5%
Average ship capacity	7,154 TEU
Utilization rate	59.2%
Average across 100 networks	
Profit	\$4.61 billion
Profit increase	8.7%
Average ship capacity	8,721 TEU
% unallocated containers (before)	5.5%
% unallocated containers (after)	5.8%
Utilization rate (before)	45.4%
Utilization rate (after)	59.6%
Execution time	12,398 seconds

the maximum attainable utilization, which we computed in Section 6.3, is only 69.47% because of the trade imbalance.

The execution time of the algorithm is 12,398 seconds, which is just over 2 minutes per network. A large portion is dedicated to the reallocation phase. Because many potential reallocation positions have to be computed, this phase is computationally quite intensive. Still, considering the results, one should not be reluctant to take the effort.

Recall that we computed the maximum attainable profit in Section 6.3.3. This so-called *upper bound* was equal to \$5.09 billion. This mixed fleet experiment yields a best profit of \$4.97 billion. That means that the relative gap with the upper bound is only 2.6%. The relative gap with the optimal solution, which incurs more costs because it represent an operational service network, will be less than 2.6%.

In summary, we see that using a mixed fleet is quite profitable. Using mixed ship sizes during the allocation phase is computationally very intensive, but as an optimization phase afterwards it is very effective and much more efficient. In future studies, we would suggest that the ship size optimization could be an extra (third) phase after the local search phase.

7.4 Varying multi-start initializations

In the benchmark study from Chapter 6 we have used specific values for the algorithmic parameters of the multi-start initialization. These parameters include the number of service networks to build, the number of routes per network, the number of ports in a string, and the fraction to randomly allocate. The values of these parameters influence the initial networks that are generated in the first phase of the multi-start local search algorithm. It is interesting to get more insight in the performance of various multi-start initializations. To that end, we present an additional experiment focusing on the number of service networks to build and the so-called subset selection technique.

Each service network that is generated in the multi-start approach, yields a different performance. This diversity of initial solutions is the advantage of the multi-start approach. When generating a large number of initial solutions that are widely spread throughout the solution space, we strive to approach the optimal solution. The more networks we build, the larger the chance of finding a better solution or obtaining the optimal solution. However, the more networks we build, the longer the execution time. The decision for the number of service networks to build is a tradeoff between effectiveness and efficiency.

In this experiment, we compare three different values for the number of service networks in a Q & 1-2 configuration. From the benchmark study, we have a case with 100 networks, and additionally we perform a case with 500 and 1,000 networks. From the three runs, we focus on the most effective network, i.e., the network with the highest profit. This experiment gives us insight in the tradeoff that is being made between effectiveness and efficiency.

The results of the experiment are presented in Table 7.9. From the results, we notice that no new best profit was found after running a 500 and 1,000 network parameter setting. Still, the $4.81 billion profit from Chapter 6, found during a run of 100 networks, is the best. We expected to find a better solution, but apparently we found an exceptionally good solution in our benchmark study. It also indicates that more networks is not always better, we do not always find a better solution, we only increase the chance of finding a better one. When we increase the number of networks, the algorithm needs more computation time. The average time needed is around 38 seconds per network. As a result, the 1,000 network run took 10.5 hours to complete. We did not register the execution time of the Q & 1-2 configuration during the benchmark study, as it was part of a batch with multiple configurations. Based on the average of 38 seconds per

Table 7.9: Best results of the different multi-start initializations

Number of service networks	100	500	1,000
Best profit	4.806b	4.773b	4.770b
Execution time (s)	N/A	19,408	37,812

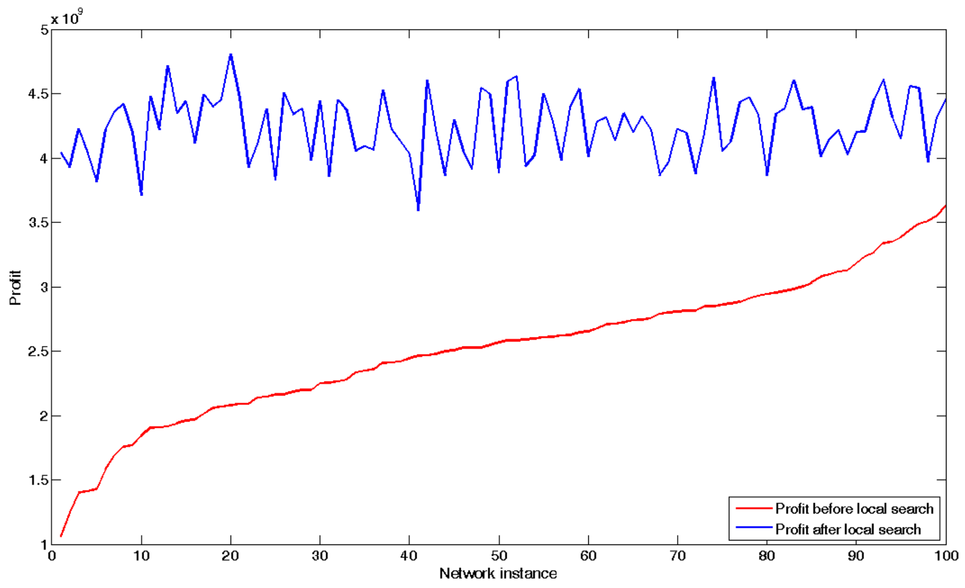
network, the 100 network run would take approximately an hour to complete. In this case, the profits from the 500 and 1,000 network run did not compensate for the extra computational time.

Not only do we want to analyze the impact of more multi-start initializations, we also want to study the so-called 'subset selection technique'. This technique is used by Brønmo et al. (2007). It works by selecting N best solutions and discard the others, before entering the local search phase. The main reason is to limit the computational time involved with local search techniques. The number of operations on each network that is required to find a local optimum is extensive. Selecting only the best solutions can dramatically speed up the process. However, the downside is that you potentially can discard good solution candidates.

To study the effect of subset selection, we plotted the profits of the initial networks, together with the profits of the same networks after the local search operators have been applied. This is done for the 100 network run with the Q & 1-2 configuration from the benchmark study. The result is visualized in Figure 7.2. Note that we sorted the networks by initial profit. When selecting a subset of best networks, for example 50%, we would only end up with the right half of the plot. However, from the figure it shows that our best solution, the \$4.81b profit in observation 20, actually results from a weak initial solution. This justifies leaving out the selection of the best initial networks in the benchmark study from Chapter 6.

In this experiment we have seen that the number of service networks to build is a tradeoff between effectiveness and efficiency. Although the results of this experiment showed differently, in general, it still holds that building more networks usually discovers networks with higher profits. This can be explained by the increased chance of finding a new best solution. It is pure luck that we obtained a better solution in the 100 network run than in the 1,000 network run. Given this, it remains difficult to decide on the number of service networks to build. The efficiency is the limiting factor in this respect. The level of efficiency needed is really a personal preference, and also depends on the computational power available. Furthermore, we showed that the subset

Figure 7.2: Network profits before and after local search



selection technique proposed in Brønmo et al. (2007), makes no sense for the routing and scheduling problem in liner shipping.

7.5 Re-applying local search operators

In Section 4.3 we introduced the operators that are used to perform local search. Each operator works in its own way, trying to improve initial solutions. Most of the operators' techniques directly affect the structure of a network, by moving, adding ports and transforming routes. Because the operators are not designed iteratively, but rather progressively, applying the same operator more than once might be useful.

We restrict this experiment to configurations which have one operator only. We extend these configurations by applying the same operator again. This prevents interference between the operators, which may affect the performance and lead to a situation where experiment results are difficult to interpret.

Using the Quantity Sort method, we choose the three configurations where each operator is applied on its own. We use the results which we have gained from the benchmark study in Chapter 6. Hence, for each of the three configurations Q & 1, Q & 2, and Q & 3, we already have 100 completed networks, including statistics about their profit and the performance increase obtained by the local search phase. In this experiment, we use each set of 100 networks and apply the same operators for a second time. We observe

Table 7.10: Results of re-applying local search operators

Configuration	Operator increase (first time)	Operator increase (second time)
Q & 1-1	74.1%	1.8%
Q & 2-2	1.5%	0.4%
Q & 3-3	15.3%	0.0%

the performance increase obtained by the re-application of the operator.

Table 7.10 shows the results of the experiment. We see that the performance increases of the second run are small, but they are present. Except for the transhipment operator, which does not yield an increase in the second run at all. When we compare operator 1 and 2, we see that the difference between the two runs is significantly larger for operator 1 than it is for operator 2. We can easily explain that difference. Each operator starts with the first route in a network, tries to improve it, and consecutively works its way through the set of routes. Once it finishes the improvement of one route, it will move forward and not look back. But operator 2 works slightly different.

Operator 2, the port-exchange operator, moves ports between pairs of routes. Hence, when it has finished optimizing one route, the operator might still change that same route at a later stage because, for example, it wants to move in a port from another route. Hence, because the route and network structure can change constantly, in the second run, the port-exchange operator can achieve relatively more than the route-length operator.

This is different for the route-length operator. This operator will try to improve one route at a time, and will never change an earlier route once it finished optimizing that route. Nevertheless, the route-length operator manages to increase profit in the second run. One might wonder why. There is a plausible explanation for this effect. The route-length operator considers every port in each route, and calculates whether this port is profitable. In other words, is there sufficient cargo from and to this port to justify the costs associated by a weekly port call? Now imagine that the operator analyzes port A and concludes that it is profitable, because it receives a lot of cargo from port B and C. Then, the operator analyzes the next port in the cycle, which is port B. This time, it concludes that it is more profitable to remove port B from the cycle. However, by removing port B, the cargo between port B and port A is lost, and so are its revenues. Sometimes this can lead to port A suddenly becoming unprofitable. But the operator does not look back to port A. In such situations, the operator will improve the solution in a second run, because it finds that port A is not profitable any more now that port B is absent. Because the route-length operator is designed to look at one port at a time, running the operator twice can further increase profit.

Finally, we see that the transhipment operator does not improve solutions during a second run. This corresponds with the way the transhipment operator works. The operator does not change the structure of the network. It merely reallocates some of the demands to other routes, by searching for shorter paths. Because the structure of the routes and the networks are not altered, it will find and use all available shorter paths in the first run, leaving nothing to do in the second run. Also, the gain of the transhipment operator is to be found in the allocation of additional demands, which could not be allocated previously because there was no direct path or there was insufficient capacity. However, all leftover demand will be allocated (if possible) in the first run. Because the operator will not re-allocate any demands in the second run, as explained before, it will also be impossible to allocate any additional demand.

In summary, we see that re-applying the same operator can be useful for the route-length operator and the port-exchange operator. Although the improvements of the second run, compared to the first run, are rather modest, one has to realize that only 1% increase in profit still represents a large sum of money. All in all, the port-exchange operator appears to benefit relatively the most from a re-application phase.

7.6 Conclusion

In this chapter, we provided additional experiments to show the multi-start local search algorithm from different angles. The experiments also justify some of the choices we made in, for example, the transhipment operator and the benchmark parameter settings.

First, we used the multi-start local search algorithm to find a solution for the seven-port data set. We adjusted the variable and parameter settings to comply with a previous study. Our current solution performs over 35% better than two alternative solutions.

Second, we compared different, alternative implementations of the transhipment operator. We introduced the single-hub single-move, and the multi-hub multi-move variant, next to the multi-hub single-move variant that was used in Chapter 6. Although one might expect that the most extensive variant, the multi-hub multi-move, performs best, we observed that the multi-hub single-move variant yields the best results. The multi-hub multi-move variant does yield higher revenues, but the additional handling costs that are incurred keeps the profit down. This supports the choice for this implementation in Chapter 4 and its use in the benchmark study from Chapter 6.

Third, we analyzed the impact of smaller ship sizes, since we observed a rather low utilization rate when using 12,000 TEU ships. In an attempt to improve utilization, we changed the ship size to 8,000 TEU. This improved the utilization rate considerably, from 45.5% to 72.4% on average. Also, the fleet size increased due to the use of smaller ships. However, the profit dropped by around 6%. This leads to the conclusion that the use of a fleet with smaller sized ships alone does not yield a better performance. Therefore, we also studied the effect of a heterogeneous fleet with mixed ship capacities. This turned out to be much more successful. We found a new best network with a profit of $4.97 billion. The average utilization rate of this network increased to 59.2%. The average ship capacity dropped to just over 7,000 TEU, which is more realistic than the 12,000 TEU we used before, given the fact that the average ship capacity of the MAE fleet is almost 8,500 TEU. With an average profit increase of nearly 9% across all 100 networks, the application of a mixed fleet is very effective.

Fourth, we changed the multi-start initializations. In Chapter 6 we used 100 networks per configuration. To see whether the use of more networks leads to a better solution, we changed the number of networks to 500 and 1,000 for the Q & 1-2 configuration. Although the use of more networks theoretically should increase the chance of finding a better solution, we found no better network using 500 or 1,000 networks. In heuristics, decisions are often based on the trade-off between performance and computation time. More (intensive) computations often lead to better solutions. However, heuristics are especially meant to limit computation time at the cost of performance. The optimal balance is difficult to determine, and also depends on the application. In our case, we did not find a better solution using more networks, but at the same time, the computational effort increased considerably. This leads to the conclusion that the use of 100 networks is reasonably good, and using 500 or even 1,000 networks does not always compensate for the extra computation time. The conclusion justifies our choice to use 100 networks in the benchmark study, especially when considering multiple algorithm configurations. Besides the number of networks, we also experimented with subset selection. By selecting a subset of best initial networks before applying the local search heuristic, one can save computational time. However, we found no convincing relation between initial profits (before local search) and final profits (after local search). Therefore, applying subset selection is likely to throw away good solution candidates and is not useful.

Finally, we studied the effect of re-applying operators. We found that the transhipment operator will not improve solutions during a second run. On the contrary, the route-length and port-exchange operator do improve solutions in the second run, although their relative increase is considerably less compared to the first run.

Chapter 8

Conclusions and Future Research

In this thesis we presented a multi-start local search algorithm for the routing and scheduling problem in liner shipping. It is a transformation of the algorithm used in Fagerholt & Lindstand (2007), that focuses on industrial and tramp shipping instead. The algorithm consists of a randomized initialization phase that generates initial networks, and a local search phase that tries to improve the solution using local search operators. The nature of the algorithm allows for variation among its two main phases. For the first phase, we implemented the *quantity sort* insertion heuristic from Brønmo et al. (2007), and proposed a *profit-driven sort* insertion heuristic. For the second phase, we proposed three new local search operators, the *route-length* operator, the *port-exchange* operator, and the *transhipment* operator. The *route-length operator* removes ports from round trips that incur more costs than revenue, and tries to allocate unassigned cargoes by adding ports to round trips. The *port-exchange operator* relocates ports within a route or between routes in an attempt to improve solutions. The *transhipment operator* introduces the use of hubs and transhipment to save costs and allocate the remaining cargoes. When combining different implementations of each phase, different algorithms are obtained. We presented a benchmark study to find out what is the most effective algorithm among these combinations. To that end, we also proposed a data set that is based on the actual Asia-Europe network of Maersk Line. Furthermore, we have presented several additional experiments to gain more insight in the working of the algorithm, the experimental design of the benchmark study, and their effect on the final result. In Section 8.1 we will present the conclusions of our research by answering the research questions. In addition, the contributions of this research are set out. Furthermore, suggestions for future research are made in Section 8.2.

8.1 Conclusions

Our main research question was *what is the performance of multi-start local search heuristics for solving routing and scheduling problems in liner shipping?* In order to answer the main research question, it was decomposed into subquestions. We have answered these eight subquestions from Section 1.3 in the previous chapters. These answers will now be summarized to answer the main research question.

1. *What are the specifics of liner shipping and what are its typical routing and scheduling problems?*

   Liner shipping is characterized by carriers providing a regular repetitive schedule of services, which is called the carrier's service network. An important problem in liner shipping is determining a service network of routes and frequencies that, for more or less given demand between ports and a fleet's capacity, maximizes profit. This problem is referred to as the routing and scheduling problem in liner shipping. In general, solution approaches are either mathematical programming oriented, which is computationally intensive but will find the optimal solution, or heuristic oriented, which is more efficient but does not guarantee to find the optimal solution.

2. *What are multi-start and local search techniques, and why do they work?*

   The multi-start technique is used to explore multiple regions of the solution space. Multi-start refers to restarting an algorithm with different parameters, such that each time a different region is explored. The main advantage of the multi-start technique is that it decreases the chance of ending up in a local optimum. Local search is a heuristic optimization technique that tries to improve initial solutions by moving through the solution space and searching for neighboring solutions that maximize a given objective. Since local search only explores a very small region of the solution space, and may get stuck in a local optimum, it very suitable to use in collaboration with the multi-start technique.

3. *Can we transform the multi-start local search algorithm from tramp shipping studies to the case of liner shipping?*

   The multi-start local search approach is a general technique, and not restricted to a specific problem. It is the implementation of the two phases that makes the algorithm applicable to a specific problem. Our starting point was research that used multi-start local search for solving the routing and scheduling problem in tramp shipping. Since the liner shipping specifics make the problem so different, it requires an alternative implementation. We were able to transform the multi-start local search algorithm from tramp shipping to liner shipping by creating such a different implementation. The most important difference between the tramp shipping and the liner shipping implementation is that tramp shipping moves with spot cargoes to find the optimal routes, whereas liner shipping mainly moves with ports to find the optimal network. We implemented two sort methods, one that was adjusted from the tramp shipping study and a new one. Furthermore, we proposed three new local search operators.

4. *What is an adequate data set to assess the effectiveness of solutions to the problem?*

   In Chapter 5, we reviewed several data sets from literature to assess their usability for our problem. We defined the required characteristics for a suitable data set as representing reality and being generally applicable. In this regard, general applicability refers to the broad usability of the data set in different problem variants. Unfortunately, the reviewed data sets were either not based on reality, or did not contain enough data to be generally applicable. As a result, we proposed a new data set that is based on Maersk's Asia-Europe service network, and strives to meet our requirements. We believe the proposed data set is adequate to assess the effectiveness of solutions to the routing and scheduling problem in liner shipping.

5. *Does the more advanced profit-driven sort insertion heuristic contribute more to the quality of the solutions in the multi-start heuristic than the simple quantity sort insertion heuristic?*

   Although the profit-driven sort method uses a profitability based scoring system to allocate demand, statistical analysis showed that there is no significant difference in network profits compared to the much more simple quantity-based sort method. However, that is only the case when comparing both sorting methods after the local search phase has been applied. When comparing the performance after the first phase of the algorithm, quantity sort yields significantly better results. However, this difference is cancelled out by the local search phase.

6. *What local search operators contribute the most to the objective function?*

   Of all three local search operators, the route-length operator yields the best profit increase. Its average improvement of nearly 76% is much higher than the 16% of the transhipment operator and the 3% of the port-exchange operator. When considering a configuration of multiple operators, the combined route-length and port-exchange operator performs best. The success of the route-length operator can be explained by its highly influential network adjustments and its profit-minded implementation. The port-exchange operator performs rather bad, probably due to limited remaining capacity on the routes, which becomes a bottleneck for relocating ports. The transhipment operator performs less well than expected, since it incurs too much costs with rerouting demands before allocating new demand pairs.

7. *What is the most effective and/or efficient multi-start local search configuration to solve routing and scheduling problems in liner shipping?*

   The most effective multi-start local search configuration is 'Q & 1-2', i.e., quantity sort in the first phase and the route-length operator followed by the port-exchange operator in the second phase. However, this configuration is not significantly better than the 'P & 1-2' and 'P/Q & 1-2' configurations, which indicates that the sorting method has no significant influence. The 'Q & 2' configuration is the most efficient, since it needs the least computation time among all configurations, namely 20 seconds for one network. In contrast, the 'Q & 1-2' configuration takes approximately 35 seconds to complete, although it is approximately 68% more effective.

8. *Under what circumstances is the use of transhipment hubs effective?*

   The use of transhipment can improve the allocation of demand, and thereby the revenue, in three ways. First, transhipment can reduce the path length from origin to destination. Second, transhipment can prevent unnecessary movements of cargo along the turning point. Third, transhipment enables unallocated demand to be allocated, since more path possibilities are available. In general, the use of transhipment hubs is effective when the revenue from the additional demands that can be transported covers the costs that are incurred as a result of transhipping the demand. Furthermore, the remaining capacity on the ships should be large enough to be able to allocate the extra demand. Only then transhipment can successfully be applied.

Finally, we are able to answer our main research question, *what is the performance of multi-start local search heuristics for solving routing and scheduling problems in liner shipping?*

We have seen that the first phase of the multi-start local search algorithm, generating the initial solutions, can not be optimized by choosing a specific sorting method. However, the second phase, improving the solutions using local search operators, does have a large influence on the final result. There are large differences in solution improvement among the local search operators. Therefore, one has to take care when choosing a specific algorithm configuration. This also goes for the input provided to the model, since the network composition and the resulting profit are sensitive to the input parameter settings. We have noticed this in the comparison of our best network with the Maersk Asia-Europe service network, where our profit deviated due to the use of a different cost structure. However, we obtained a better judgement of the performance of the multi-start local search technique by computing the upper bound and the relative gap to our best solution. The assessment of the performance of the multi-start local search algorithm remains subjective. We believe the relative gap to the upper bound is a good measure in this respect. Considering that the optimal solution will be lower than the upper bound, although we do not know by how much, a relative gap to the upper bound of less than 10% seems acceptable. Given that we obtained a relative gap of 5.5%, and the answers to the subquestions above, we conclude that multi-start local search heuristics, as a technique, perform satisfactory when applied to liner shipping routing and scheduling problems.

The problem of routing and scheduling in liner shipping has received little attention in literature. Most research focused on mathematical programming, and the research focusing on heuristics applied the problem to industrial and tramp shipping instead. Therefore, this paper has a large contribution to this field, especially with respect to the following points.

- Multi-start local search heuristics are proposed for liner shipping operations, these provide adequate solutions to the routing and scheduling problem in liner shipping.
- A benchmark of the various multi-start local search heuristics is added, with an analysis specifying the most effective configuration.
- A new data set that combines general applicability and reality is proposed, which is based on the actual Maersk Asia-Europe service network.

8.2 Future Research

This research also uncovers possible future research directions in the field of routing and scheduling for liner shipping.

First, future work could focus on improving the studied algorithm by creating additional local search operators. Apart from the route-length, port-exchange, and transhipment operator, alternative operators might yield good results. Additionally, improving the current operators, especially the transhipment operator, might also be worthwhile. The transhipment operator can be improved by introducing a revenue/cost tradeoff for transhipment moves. Moreover, rerouting already allocated demands to gain capacity should only be done when the space is really used. The operator can also be changed to adjust the network, rather than purely focusing on the allocation of demand, although the network composition analysis in Section 6.4.2 indicated that the profit benefits the most from improving allocation.

Second, future work might focus on different approaches for solving the routing and scheduling problem in liner shipping. In this thesis, we studied the heuristic multi-start local search approach. We believe there are still many opportunities with heuristic algorithms, although the solution will likely remain suboptimal. In this respect, it is also interesting to gain insight in the performance of metaheuristic approaches for solving the problem. These might deliver solutions that are closer to the optimal solution. Since there is little literature available that focuses on solving the problem in liner shipping operations, several approaches can still be applied.

Third, future work can discover different problem formulations. The problem can easily be extended by leaving out some simplifying assumptions. For example, one can take into account the feedering of intra-regional demand or use a restricted fleet. In addition, some variables in our problem formulation could be improved to approach reality. For example, in reality the fuel consumption depends on the cruising speed and the cargo load of the ship. Furthermore, the operational cost that we used in the additional experiment in Section 7.3 could be made dependent of the ship's capacity. One can also follow the trends in liner shipping to obtain research directions. Lately, the bunkering of oil is in the middle of attention. When introducing bunkering to the problem, an additional objective becomes determining the best spots to bunker for fuel. However, the characteristic of liner shipping is that it follows a strict schedule, which already puts some restrictions on the bunkering spots. Another trend is slow steaming, which means the vessels travel at low speed to save costs. It might be interesting to extend the problem with a flexible speed variable to study the effect of different speeds. Lastly, the problem can be extended with the empty container repositioning problem. This problem is a major cost-driver for carriers (Veenstra 2005), and is an important step in approaching reality. It may be clear, every change in assumptions creates a new variant of the problem, and there are numerous variants and extensions possible.

Chapter 9

Discussion

The experimental design and the findings of this research instigate some further discussion regarding several points. First, we want to discuss the implementation of the transhipment operator, that did not perform as expected. Second, we want to focus on the influence of the fleet specification on the experimental design of this study, especially regarding the surprising result of the mixed fleet approach. Last, we want to discuss the findings regarding the subset selection technique, that seems to contradict previous work from literature.

One can argue whether the transhipment operator should have been implemented differently. The average improvement of the transhipment operator to the initial solutions were rather disappointing. Moreover, combining the transhipment operator with another operator in a single configuration resulted in even worse improvements. In the current implementation, the reallocation of demand to gain space, and the allocation of remaining demands are split in two separate steps. Furthermore, there is no real cost-benefit tradeoff calculation being performed when (re)allocating demands. Anyway, we did not expect to reallocate so much demand, since we estimated to have higher utilization rates than we ultimately did. It turned out that the first step, the reallocation of demands, incurs too much costs. Afterwards, it had not even been necessary to reallocate those demands, since the additional demand that can be allocated does not cover for the costs.

The rationale behind the two-step design was the idea that sometimes you need to make costly changes, in order to obtain a better solution than we were able to obtain before. In other words, we wanted to prevent ending up in a local optimum, and go for a better optimum. However, we acknowledge that this is not the task of the local search operator. The local search operator should find the local optimum, and the multi-start characteristic is responsible for a diverse set of these local optima, such that we can pick the best solution.

Currently, the transhipment operator only focuses on the allocation of demands, and does not adjust the service network itself like the other operators. On the one hand, this seems no big deal, since we know from the network composition analysis in Section 6.4.2 that the allocation is very important for the final profit. On the other hand, it might be possible to obtain better results when the operator adjusts the network, since it will provide new allocation opportunities.

In the problem formulation of Chapter 3 we specified the precise assumptions that we made, in order to scope the research. We made different kinds of assumptions, either representing reality, extending reality, or delimiting the scope of the study. One of the assumptions extending reality was that we assumed a fleet with an unrestricted size, consisting of vessels with similar size and capacity. Accordingly, in the experimental design of the benchmark study, we specified a ship capacity of 12,000 TEU. So far, so good. However, in the analysis of our best network, we found a rather low average utilization rate of the routes. Nevertheless, we experienced that many routes had one or more legs with nearly full utilization. This effect was clearly observable in Figure 6.3. It made us wonder what ship size would be optimal to use. To that end, we performed the additional experiment in Section 7.3.

First, we studied the effect of having a lower ship capacity, while still considering a homogeneous fleet. We performed an experiment, running the algorithm with a ship capacity of 8,000 TEU. Results indicated that both the average profits and the best profit of the 8,000 TEU run underperformed against the 12,000 TEU run, although the utilization rate increased. Next, we performed an experiment that ignored the assumption of a homogeneous fleet, and allowing a mixed fleet. The starting point was a set of networks from the benchmark study, no new networks were created. We removed some demands from the high-utilization legs of each route in order to decrease the ship size. We also tried to reallocate that demand to other routes. In this way, every route obtained ships with a different size, together forming a mixed fleet. The results of this experiment showed a surprising improvement in profit. It turned out that a mixed fleet of smaller ships can be very profitable.

One can argue whether the initial choice for a 12,000 TEU ship capacity was sensible, especially when considering the average capacity of around 8,500 TEU that Maersk uses. We defended this choice by pointing at the trend of larger container ships, and avoiding the computational intensity of mixed ship capacities. However, the difference of 3,500 TEU is still very large. Afterwards, it seemed that the steps performed in the additional experiment could well be used as a third phase to the multi-start local search algorithm. The profit increases among the networks were different, meaning that the best solution from the local search phase may be outperformed by another solution after applying the third phase. The same thing currently happens in the second phase of the algorithm, where the best networks from phase one are not necessarily the best networks after phase 2 has been applied. Therefore, it fits pretty good in the current implementation. Actually, extending the algorithm with this third phase may be even better than sticking to a homogeneous fleet. The reason for this is that the extension of the algorithm would yield a heterogeneous fleet, and such a mixed fleet was showed to be more profitable.

In the additional experiment presented in Section 7.4, we focused on varying multi-start initializations. As part of this experiment, we studied the effect of the subset selection technique, that is used in the multi-start local search algorithm for tramp shipping in Brønmo et al. (2007). In their study, the technique is part of the first phase of the algorithm, that generates the initial solutions. It works by selecting N best solutions and discarding the others, before entering the local search phase. The rationale for using this technique is to limit the computational time by only continuing with the best initial solutions.

Nevertheless, the result of our experiment showed that the subset selection technique does not make sense, at least for liner shipping operations. We found that there is no direct relation between the quality of the initial solution, and its quality after the local search phase has been applied. Moreover, our optimal solution was obtained from a very weak initial solution, that definitely had been discarded if we used the subset selection technique. The question is whether the difference in shipping type, i.e., tramp shipping versus liner shipping, has such an influence that this technique becomes useless. We believe this is not the case. The rationale of limiting the computation time is no argument for using this approach, since creating less networks eventually yields the same result. Therefore, the advantages of its application in Brønmo et al. (2007) remain unclear. At least, because the rationale is not trivial, the previous study should have clarified the characteristics of this technique that make it work in the case of tramp shipping operations.

Glossary

Cycle	A round trip based on the *string of ports.*
Direct path	Used to indicate that the origin and destination of a demand lay in a single route. Then, the direct path is the part of the route from origin to destination.
Feeder-line	Provides regional services on *feeder* routes, feeding and distributing containers to and from the hub using smaller-sized ships like barges.
Frequency	The number of departures from a port for a specific *service* in a given time frame.
Hub	A *transhipment* port where cargo is transferred from one ship to another, to reach its destination. The transhipment can take place either directly between the ships, or after short-term storage in the container terminal.
Leg	The route from a port to the next port in the string of ports.
Loop	The same as a *cycle.*
Main-line	Liner shipping route, connecting the largest ports (i.e., regional hubs) with big container ships.
OD	Abbreviation of origin-destination.
Path	The ports that a demand comes across when traveling from origin to destination. Also used to refer to either a *direct path* or *transhipment path.*
Pendulum	The same as a *cycle.*

Rerouting	Removing allocated demand from a origin-destination path, in order to allocate to another path. The advantages of rerouting can be gaining ship capacity on a route, or incurring less costs, for example by visiting less ports.
Route	A part of or the full *string of ports*, that the ship visits to deliver a cargo. Used when referring to both the complete cycle, i.e. the full string of ports, as well as a part of the cycle, when considering the origin and destination of a cargo.
Service	The same as a *cycle*. Sometimes also used to refer to a part of the cycle.
Service network	The set of *services* that a carrier provides.
Slack	The time between the duration of a round trip and the next integer number of weeks, when the cycle starts again.
String (of ports)	The sequence of ports that a ship visits to complete a cycle.
TEU	Twenty-foot equivalent unit, one TEU refers to a 20-feet long container. The number of containers is measured in TEU.
Throughput	The total number of movements into or out of a port, measured in TEU.
Transhipment	Transferring cargo from one ship to another, in order to continue the journey to its destination. With transhipment, different services are combined to provide the transport from the origin to the destination.
Transhipment path	The full path from the origin to the destination of a demand, when the demand is transhipped one or multiple times. As a result, a transhipment path can travel via several routes.
Trip	The same as a *route*.
Voyage	The same as a *cycle*.

Bibliography

Agarwal, R. & Ergun, O. (2008), 'Ship scheduling and network design for cargo routing in liner shipping', *Transportation Science* **42**(2), 175–196.

Alvarez, J. (2009), 'Joint routing and deployment of a fleet of container vessels', *Maritime Economics & Logistics* **11**(2), 186–208.

Ambrosino, D., Anghinolfi, D., Paolucci, M. & Sciomachen, A. (2009), 'A new three-step heuristic for the master bay plan problem', *Maritime Economics & Logistics* **11**(1), 98–120.

A.P. Moller - Maersk Group (2009), 'Annual Report 2009'.

A.P. Moller - Maersk Group (2010), 'Interim Report 2010'.

Armitage, P. & Berry, G. (1994), *Statistical methods in medical research*, 3 edn, Blackwell Science Ltd, Massachusetts, USA.

Aversa, R., Botter, R., Haralambides, H. & Yoshizaki, H. (2005), 'A mixed integer programming model on the location of a hub port in the east coast of South America', *Maritime Economics & Logistics* **7**(1), 1–18.

Baird, A. (2006), 'Optimising the container transhipment hub location in northern Europe', *Journal of Transport Geography* **14**(3), 195–214.

Bodin, L., Golden, B., Assad, A. & Ball, M. (1983), 'Routing and scheduling of vehicles and crews: The state of the art', *Computers & Operations Research* **10**(2), 63–211.

Bräysy, O. & Gendreau, M. (2005), 'Vehicle routing problem with time windows, part ii: Metaheuristics', *Transportation Science* **39**(1), 119–140.

Brønmo, G., Christiansen, M., Fagerholt, K. & Nygreen, B. (2007), 'A multi-start local search heuristic for ship scheduling–a computational study', *Computers & Operations Research* **34**(3), 900 –917.

Brønmo, G. & Løkketangen, A. (2007), An adaptive constructive solution generator for ship scheduling DSS. Presented at NIK 2007 conference, Oslo.

Christiansen, M., Fagerholt, K., Hasle, G., Minsaas, A. & Nygreen, B. (2009), 'Maritime transport optimization: An ocean of opportunities', *OR/MS Today* **36**(2), 26–31.

Christiansen, M., Fagerholt, K. & Ronen, D. (2004), 'Ship routing and scheduling: Status and perspectives', *Transportation Science* **38**(1), 1–118.

Containerisation International (2010), `http://www.ci-online.co.uk/`. News and information service for the container industry with online data sources. Retrieved July 20, 2010.

Cordeau, J., Toth, P. & Vigo, D. (1998), 'A survey of optimization models for train routing and scheduling.', *Transportation Science* **32**(4), 380–405.

Cullinane, K., Khanna, M. & Song, D. (1999), How big is beautiful: Economies of scale and the optimal size of containership, *in* 'Liner Shipping: Whats Next?', Proceedings of the 1999 IAME conference, Halifax., pp. 108–140.

Dantzig, G. & Ramser, J. (1959), 'The truck dispatching problem', *Management Science* **6**(1), 80–91.

De Souza Junior, G., Beresford, A. & Pettit, S. (2003), 'Liner shipping companies and terminal operators: Internationalisation or globalisation?', *Maritime Economics & Logistics* **5**, 393–412.

Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M. & Soumis, F. (1997), 'Daily aircraft routing and scheduling', *Management Science* **43**(6), 841–855.

Distances.com Distance Calculcator (2010), 'World Ports Distances', `http://www.distances.com/`. Retrieved July 20, 2010.

Dong, J. & Song, D. (2009), 'Container fleet sizing and empty repositioning in liner shipping systems', *Transportation Research Part E: Logistics and Transportation Review* **45**(6), 860–877.

Drewry Shipping Consultants (2007), Annual Container Market Review & Forecast 2006/07, Technical report, London.

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

Fagerholt, K. (2004), 'Designing optimal routes in a liner shipping problem', *Maritime Policy & Management* **31**(4), 259–268.

Fagerholt, K. & Lindstand, H. (2007), 'Turborouter: An interactive optimisation-based decision support systems for ship routing and scheduling', *Maritime Economics & Logistics* **9**(3), 214–233.

Farrar, D. & Glauber, R. (1967), 'Multicollinearity in regression analysis: The problem revisited', *The Review of Economics and Statistics* **49**(1), 92–107.

Feng, C. & Chang, C. (2008), 'Empty container reposition planning for intra-Asia liner shipping', *Maritime Policy & Management* **35**(5), 469–489.

Ferrari, C., Parola, F. & Benacchio, M. (2008), 'Network economies in liner shipping: the role of home markets', *Maritime Policy & Management* **35**(2), 127–143.

Fusillo, M. (2004), 'Is liner shipping supply fixed?', *Maritime Economics & Logistics* **6**, 220–235.

Fusillo, M. (2009), 'Structural factors underlying mergers and acquisitions in liner shipping', *Maritime Economics & Logistics* **11**, 209–226.

Heaver, T., Meersman, H., Moglia, F. & van de Voorde, E. (2000), 'Do mergers and alliances influence european shipping and port competition?', *Maritime Policy & Management* **27**(4), 363–373.

Hothorn, T., Leisch, F., Zeileis, A. & Hornik, K. (2005), 'The design and analysis of benchmark experiments', *Journal of Computational and Graphical Statistics* **14**(3), 675–699.

Imai, A., Nishimura, E., Papadimitriou, S. & Liu, M. (2006), 'The economic viability of container mega-ships', *Transportation Research Part E* **42**, 21–41.

Lachner, S. & Boskamp, V. (2010), Routing and scheduling in liner shipping: two heuristic algorithms. Erasmus University Rotterdam, The Netherlands, Master Seminar.

Lam, J. (2009), 'An integrated approach for port selection, ship scheduling and financial analysis', *Netnomics* . Available online: `http://www.springerlink.com/content/x26gp1143qm47714/`.

Laporte, G., Gendreau, M., Potvin, J. & Semet, F. (2000), 'Classical and modern heuristics for the vehicle routing problem', *International Transactions in Operational Research* **7**(4-5), 285–300.

Lawrence, S. (1972), *International Sea Transport: The Years Ahead*, Lexington, MA: Lexington Books.

Madsen, O. B., Ravn, H. F. & Moberg Rygaard, J. (1995), 'A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives', *Annals of Operations Research* **60**, 193–208.

Maersk Line (2010), 'Asia - Europe service network', `http://www.maerskline.com/link/?page=brochure&path=/routemaps/newnetwork/asiaeur`. Retrieved June 12, 2010.

Maersk Line BAF (2010), `http://baf.maerskline.com/`.

MAN B& W Diesel A/S (2008), 'Propulsion Trends in Container Vessels'.

Man, K. (2007), Routing and scheduling container liners between Asia and Europe. Erasmus University Rotterdam, the Netherlands, Bachelor Thesis.

Marti, R. (2003), *Multi-Start Methods*, Springer New York, chapter 12, pp. 355–368.

Midoro, R. & Pitto, A. (2000), 'A critical evaluation of strategic alliances in liner shipping', *Maritime Policy & Management* **27**(1), 31–40.

Mitsui O.S.K. Lines (2010), 'AEX service as of May 2010', `http://www.molpower.com/VLCWeb/UIStatic/services/service_aex.aspx`. Retrieved May 22, 2010.

Morton, R. (2007), 'Ocean shipping steaming ahead', Logistics Today. Interview with President of NYK Line.

Naso, D., Surico, M., Turchiano, B. & Kaymak, U. (2007), 'Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete', *European Journal of Operational Research* **177**(3), 2069–2099.

Nicolai, R. & Dekker, R. (2009), 'Automated response surface methodology for simulation optimization models with unknown variance', *Quality Technology & Quantitative Management* **6**(3), 325–352.

Notteboom, T. (2004), 'Container shipping and ports: An overview', *Review of Network Economics* **3**(2), 86–106.

Notteboom, T. (2006), 'The time factor in liner shipping services', *Maritime Economics & Logistics* **8**, 19–39.

Notteboom, T. & Rodrigue, J. (2008), 'Containerisation, box logistics and global supply chains: The integration of ports and liner shipping networks', *Maritime Economics & Logistics* **10**, 152–174.

Notteboom, T. & Vernimmen, B. (2009), 'The effect of high fuel costs on liner service configuration in container shipping', *Journal of Transport Geography* **17**, 325–337.

PortWorld Distance Calculcator (2010), 'Ship Voyage Distance Calculator', `http://www.portworld.com/map`. Retrieved July 20, 2010.

Potvin, J. (2009), 'Evolutionary algorithms for vehicle routing', *INFORMS Journal on Computing* **21**(4), 518–548.

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

Ronen, D. (1986), 'Short-term scheduling of vessels for shipping bulk or semi-bulk commodities originating in a single area', *Operations Research* **34**(1), 164–173.

Ronen, D. (2011), 'The effect of oil price on containership speed and fleet size', *Journal of the Operational Research Society* **62**(1), 211–216.

Sea Rates Distance Calculcator (2010), 'Port to Port Distances', `http://www.searates.com/reference/portdistance/`. Retrieved July 20, 2010.

Shintani, K., Imai, A., Nishimura, E. & Papadimitriou, S. (2007), 'The container shipping network design problem with empty container repositioning', *Transportation Research Part E* **43**, 39–59.

Song, D. & Carter, J. (2009), 'Empty container repositioning in liner shipping', *Maritime Policy & Management* **36**(4), 291–307.

Ting, S. & Tzeng, G. (2003), 'Ship scheduling and cost analysis for route planning in liner shipping', *Maritime Economics & Logistics* **5**(4), 378–392.

Toth, P. & Vigo, D., eds (2001), *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

TurboRouter (2007), `http://www.sintef.no/Projectweb/TurboRouter/`.

UNCTAD (2009), ‘United nations conference on trade and development review of maritime transport 2009’. Geneva.

Van de Weerd, V. (2009), Investigate the ship routing and cargo allocation in container liner shipping between Europe and Asia. Erasmus University Rotterdam, the Netherlands, Bachelor Thesis.

Van der Meer, G. (2009), Scheduling container liners between Asia and Europe with different speeds. Erasmus University Rotterdam, the Netherlands, Bachelor Thesis.

Van der Meer, G. (2011), Introducing shuttle services in schedules for container liners between Asia and Europe. Erasmus University Rotterdam, the Netherlands, Master Thesis.

Veenstra, A. (2005), *Managing Closed-Loop Supply Chains*, Vol. 3, Springer Berlin Heidelberg, chapter 6, pp. 65–76.

Yan, S., Chen, C. & Lin, S. (2009), ‘Ship scheduling and container shipment planning for liners in short-term operations’, *Journal of Marine Science and Technology* **14**(4), 417–435.

Appendix

Appendix A

Algorithms

In this appendix chapter we present pseudocode formulations of the multi-start local search algorithm. Originally, the algorithm is implemented in Matlab code, but displaying the full Matlab codes required numerous pages. Therefore, the pseudocode presents a compact version of the Matlab code, in order to provide the reader with the general message of the algorithm. Table A.1 gives an overview of the algorithms in this appendix, each representing a different part of the multi-start local search heuristic.

Table A.1: Overview of algorithm pseudocodes

Algorithm	Description
2	The Multi-Start Local Search Algorithm
3	Generate a random route
4	Allocate random demand
5	Apply sorting method
6	Select and clean the best networks
7	Apply local search operators
8	Local Search: the route-length operator
9	Local Search: the intra-route port exchange operator
10	Local Search: the inter-route port exchange operator
11	Local Search: the transhipment operator
12	Transhipment operator: reroute allocated demand
13	Transhipment operator: allocate remaining OD demand
14	Transhipment operator: get transhipment path candidates
15	Single-hub single-move (SHSM) transhipment operator
16	Multi-hub multi-move (MHMM) transhipment operator
17	MHMM transhipment operator: get transhipment path candidates

Algorithm 2: The Multi-Start Local Search Algorithm

```
Input: A set of ports $P$ with size $m$, an origin-destination matrix $Q$ with size $m \times m$, an
       inter-port distance matrix D with size $m \times m$
Output: A (sub)optimal service network $S$
var $networks$ = [];
for $i \leftarrow 1$ to 100 do
    var $n$ = new Network();
    $n.Q = Q$;
    for $j \leftarrow 1$ to $randint(5,20)$ do // random number of routes
        var $route$ = generateRandomRoute($P$, $Q$) ;                     // See Alg. 3
        $n$.append($route$);
        for $k \leftarrow 1$ to 3 do allocateRandom($route$, $n.Q$) ;          // See Alg. 4
    end
    $networks$.append($n$);
end
foreach element $n$ of $networks$ do
    var $cargoes$ = matrix2array($n.Q$) ; // convert OD matrix to array
    var $method$ = random(Quantity || PDA);
    if $method$ == 'Quantity' then sort($cargoes$, descending by quantity);
    foreach element $c$ of $cargoes$ do
        var $origin = c.origin$; var $destination = c.destination$;
        var $feasibleroutes$ = [];
        foreach element $route$ of $n.routes$ do
            if isElement(origin, route) && isElement(destination, route) then
            $feasibleroutes$.append($route$);
        end
        var $routes$ = applySorting($method$, $c$, $feasibleroutes$) ;         // See Alg. 5
        var $j = 1$;
        while $q.demand > 0$ && $j <$ count(routes) do
            // take minimum of either demand or remaining capacity
            var $demand$ = min( $q.demand$, $routes(j).rc$ );
            allocateDemand( $routes(j)$, $q.origin$, $q.destination$, $demand$ );
            $j = j + 1$; $q = q$ - $demand$;
        end
    end
end
$networks$ = selectCleanTopNetworks($networks$) ;                                  // See Alg. 6
foreach network $n$ of $networks$ do applyLocalSearch($n$);                        // See Alg. 7
sort($networks$, $-profit$) ; // sort descending by column profit
var $S = networks(1)$ ; // select the best network
```

Algorithm 3: Generate a random route

```
Input: A set of ports $P$ with size $m$, an origin-destination matrix $Q$ with size $m \times m$
Output: A random route $r$
foreach element $p$ of $P$ do
    // Remove ports that don't have demand
    if ( sum[Q(p,:)] + sum[Q(:,p)] == 0 ) then $P$.remove($p$);
end
var stringlength = randint(2, (2*m)-2);
var feasible = false;
while feasible == false do
    var string = [];
    for $i \leftarrow 1$ to stringlength do string.append( randint(1,$m$) );
    sort(string, ascending);
    // get vector with number of times each element of string occurs
    var $c$ = histcount(string);
    if c(1) == 1 && c(end) == 1 && find(c > 2)==false then
        feasible = true;
    end
end
var $u$ = unique(string);
// first store the duplicates
var eastbound = u( find(c==2) );
var westbound = u( find(c==2) );
// then randomly assign the single occurences
var singles = u( find(c==1) );
singles( [1 end] ) = [] ;                    // remove the starting and turning point
for $i \leftarrow 1$ to length(singles) do
    var $j$ = randint(1,2);
    if j==1 then eastbound.append(singles(i));
    if j==2 then westbound.append(singles(i));
end
// finally, assign the starting and turning point
eastbound.append( u(1) );
westbound.append( u(end) );
// sort the strings
sort(eastbound, ascending);
sort(westbound, desending);
var $r$ = vertcat(eastbound, westbound);
return $r$;
```

Algorithm 4: Allocate random demand

```
Input: A route $route$, an origin-destination matrix $Q$ with size $m \times m$
var $demand = 0$;
while $demand == 0$ do
    var $origin$ = randint(1,$m$);
    var $destination$ = randint(1,$m$);
    $demand = Q(origin,destination)$;
end
var $fraction$ = 0.01 * randint(5,15);
$demand = demand * fraction$;
// Calculate remaining capacity on the path
var $rc$ = calculateRC($route$, $origin$, $destination$);
if $rc > demand$ then
    allocateDemand($route$, $origin$, $destination$, $demand$);
    updateRCMatrix($origin$, $destination$, $demand$);
    $Q(origin, destination)$ -= $demand$;
end
```

Algorithm 5: Apply sorting method

```
Input: Sorting method $method$, specific cargo $c$, list of feasible routes $feasibleroutes$
Output: Set of ordered routes $routes$
var $routes = feasibleroutes$;
var $origin = c.origin$; var $destination = c.destination$;
if $method$ == 'Quantity' then
    foreach element route of routes do $route.rc$ = calculateRC($route$, $origin$, $destination$);
    sort($routes$, $-rc$) ; // sort decreasingly by remaining capacity
else if $method$ == 'PDA' then
    foreach element route of routes do
        $route.numports$ = findShortestPath($route$, $origin$, $destination$);
        $route.activeness$ = findActivePorts($route$, $origin$, $destination$);
        var $utilCurrent$ = calcCurrentUtil($route$);
        var $utilNew$ = calcNewUtil($route$, $origin$, $destination$, $c.quantity$);
        $route.utilDiff$ = ($utilNew$ - $utilCurrent$) / $utilCurrent$;
    end
    // normalize each attribute using the max-min method
    maxminnorm($routes.numports$, $routes.activeness$, $routes.utilDiff$);
    foreach element r of routes do
        $r.score$ = sum($r.numports$, $r.activeness$, $r.utilDiff$);
    end
    sort($routes$, $-score$) ; // sort decreasingly by score
end
return $routes$;
```

Algorithm 6: Select and clean the best networks

```
Input: A set of networks $networks$
Output: A set of cleaned, best networks $networks$
foreach $network\ n$ of $networks$ do $n.profit$ = calculateProfit($n$);
sort($networks$, $-profit$) ; // sort the networks descending by profit
if $count(networks) > 200$ then var $selection$ = 0.15 * count($networks$);
else if $count(networks) > 50$ then var $selection$ = 0.3 * count($networks$);
else var $selection$ = 0.6 * count($networks$);
// selection of the best networks
$networks$ = $networks$(1:$selection$);
// clean inactive ports, i.e.  ports without (off)loading movements
foreach $network\ n$ of $networks$ do
    foreach $route\ r$ of $n.routes$ do removeInactivePorts($r$);
end
return $networks$;
```

Algorithm 7: Apply local search operators

```
Input: A network $n$
Output: A network $n$
// Run Route-Length Operator
$n$ = $routeLengthOperator(n)$ ;                                  // See Alg.  8
// Run Port-Exchange Operator
$n$ = $portExchangeOperator\_IntraPortExchange(n)$ ;              // See Alg.  9
$n$ = $portExchangeOperator\_InterPortExchange(n)$ ;              // See Alg.  10
// Run Transhipment Operator
$n$ = $transhipmentOperator(n)$ ;                                 // See Alg.  11
return $n$;
```

Algorithm 8: Local Search: the route-length operator

```
Input: Network $n$, interport distance matrix $D$, cost constants $costs$, and sailing speed $speed$
Output: A network $n$ with the route-length operator applied
foreach route $r$ of $n.routes$ do          /* first part: remove cost-inefficient ports */
    for $i \leftarrow 1$ to $length(r.String)$ do
        var $port = r.String(i)$;
        var $costs = 0$;
        $costs = costs + costs.port$ ; // add the fixed costs per port
        var $p\_port = r.String(pos-1)$ ; // get the port before $port$
        var $n\_port = r.String(pos+1)$ ; // get the port after $port$
        var $distance\_diff = ( D(p\_port, port) + D(port, n\_port) ) - D(p\_port, n\_port)$;
        $costs = costs + (distance\_diff * costs.fuel\_per\_mile)$ ; // add fuel costs per mile
        var $quantity$ = getQuantityAtPort($port$);
        $costs = costs + (quantity * costs.handling)$ ; // add the handling costs per TEU
        // - add the costs for addtional vessels
        var $sailingweeks\_before$ = ceil( (getTotalMileage($r$) / ($speed$ * 24)) / 7 );
        var $r2$ = removePortFromRoute($r$, $port$);
        var $sailingweeks\_after$ = ceil( (getTotalMileage($r2$) / ($speed$ * 24)) / 7 );
        var $costs\_per\_vessel = costs.capital + (costs.capital * cost.operational)$;
        $costs = costs + ( (sailingweeks\_after - sailingweeks\_before) * costs\_per\_vessel )$;
        var $revenue$ = getRevenueFromPort($r$, $port$);
        if $costs > revenue$ then $r$ = removePortFromRoute($r$, $port$);
    end
    storeRoute($n$, $r$);
end
for $i \leftarrow 1$ to $count(n.OD\_matrix)$ do          /* second part: add ports */
    for $j \leftarrow 1$ to $count(n.OD\_matrix)$ do
        $demand = n.OD\_matrix(i,j)$;
        if $demand > 0$ then
            var $profits$ = [];
            for $k \leftarrow 1$ to $count(n.routes)$ do
                $r = n.routes(k)$;
                $r2$ = allocateDemand($r$, $i$, $j$, $demand$);
                var $gain$ = calculateProfit($r2$) $-$ $calculateProfit$(r);
                var $maxQuantity$ = calcMaxQuantity($r2$, $i$, $j$);
                if $gain > 0$ then
                    $profits$.append($k$, $gain$, $maxQuantity$);
                end
            end
            $profits$ = sort($profits$, -profit) ; // sort decreasingly by profit
            var $m = 1$;
            while $demand > 0$ && $m <= count(profits)$ do
                var $routeindex = profits(m,1)$;
                var $quantity = profits(m, 3)$;
                $n.routes(routeindex)$ = allocateDemand($n.routes(routeindex)$, $i$, $j$, $quantity$);
                $demand = demand - quantity$;
                $m = m + 1$;
            end
            $n.OD\_matrix(i,j) = demand$;
        end
    end
end
return $n$;
```

Algorithm 9: Local Search: the intra-route port exchange operator

```
Input: A network $n$
Output: A network $n$
for $i \leftarrow 1$ to $count(n.routes)$ do
    var $route = n.routes(i)$;
    var $string = route.String$;
    var $singleports$ = getSinglePorts($string$);
    $singleports$([1 end]) = [] ; // remove turning points
    foreach element $port$ of $singleports$ do
        $leftpart = string$(1:max($string$)-1);
        $rightpart = string$(max($string$):end);
        if $isElement(port, leftpart)$ then
            var $pathlength1$ = getTotalPathLength($route$);
            var $route2 = route$;
            // move the port in the string from the left part to the right part
            $route2.String$ = cat( $leftpart$(-$port$), $rightpart$(1:find($rightpart > port$, 'last')),
            $port$, $rightpart$(find($rightpart < port$, 'first'):end) );
        end
        else if $isElement(port, rightpart)$ then
            var $pathlength1$ = getTotalPathLength($route$);
            var $route2 = route$;
            // move the port in the string from the right part to the left part
            $route2.String$ = cat( $rightpart$(-$port$), $leftpart$(1:find($leftpart < port$, 'last')),
            $port$, $leftpart$(find($leftpart > port$, 'first'):end) );
        end
        $route2.RC$ = rebuildRCMatrix($route2$);
        // check if the new string is feasible
        if $min(route2.RC) >= 0$ then
            // check if the new string is sensible
            $pathlength2$ = getTotalPathLength($route2$);
            if $pathlength1 > pathlength2$ then
                $n.routes(i) = route2$;
                $route = route2$;
                $string = route.String$;
                /* Try to allocate new demand                                */
                for $k \leftarrow 1$ to $count(n.OD\_matrix)$ do
                    for $l \leftarrow 1$ to $count(n.OD\_matrix)$ do
                        var $demand = n.OD\_matrix(k,l)$;
                        if $demand > 0$ then
                            var $q$ = maxQuantity($route$, $k$, $l$);
                            $q$ = min($q$, $demand$);
                            allocateDemand($route$, $k$, $l$, $q$);
                            $n.OD\_matrix(k,l) = n.OD\_matrix(k,l) - q$;
                        end
                    end
                end
            end
        end
    end
end
return $n$;
```

Algorithm 10: Local Search: the inter-route port exchange operator

```
Input: A network $n$
Output: A network $n$
var $combinations$ = $nchoosek$(1:size($n.routeset$,2), 2) ; // obtain all possible route pairs
for $i \leftarrow 1$ to $size(combinations, 1)$ do
    var $route1$ = $n.routeset$( $combinations(i, 1)$ );
    var $route2$ = $n.routeset$( $combinations(i, 2)$ );
    foreach $element\ port\ of\ route1.String$ do
        // obtain all counterpart ports
        var $cp$ = unique([$route1.Destinations(route1.Origins{==}port)$ ... ...
        $route1.Origins(route1.Destinations{==}port)$]);
        // check if all counterpart ports exist in the second route
        if $min(ismember(cp, route2.String)) > 0$ then
            var $profit1$ = calculateProfit($n$);
            // option 1: merge the ports
            if $ismember(port, route2.String)$ then
                var $n2$ = mergePortToRoute($n$, $route1$) ; // merge to route1
                var $profit2$ = calculateProfit($n2$);
                var $n3$ = mergePortToRoute($n$, $route2$) ; // merge to route2
                var $profit3$ = calculateProfit($n3$);
                // save the network with the highest profit
                if $profit3 > profit2$ && $profit3 > profit1$ then
                    $n = n3$;
                end
                else if $profit2 > profit3$ && $profit2 > profit1$ then
                    $n = n2$;
                end
            end
            // option 2: move the port to route2
            else
                var $string = n.routeset(route2).String$;
                var $eastbound = string(1 : find(string == max(string)) - 1)$;
                var $westbound = string(numel(eastbound) + 1 : end)$;
                deletePort($n.routeset(route1)$, $port$);
                var $n2$, $n3 = n$;
                $n2.routeset(route2).String$ = cat(insertPort($eastbound$, $port$), $westbound$ );
                $n3.routeset(route2).String$ = cat($eastbound$, insertPort($westbound$, $port$) );
                $profit2$ = calculateProfit($n2$);
                $profit3$ = calculateProfit($n3$);
                // save the network with the highest profit: analogous to above
            end
        end
    end
end
return $n$;
```

Algorithm 11: Local Search: the transhipment operator

```
   Input: A network $n$
   Output: A network $n$
   // Re-route allocated OD pairs via transhipment hub(s).
1  $n = rerouteExisting(n)$ ;                                    // See Alg. 12
   // Allocate remaining OD matrix demand via transhipment hub(s).
2  $n = AllocateLeftOvers(n)$ ;                                  // See Alg. 13
3  return $n$;
```

Algorithm 12: Transhipment operator: reroute allocated demand

```
Input: A network $n$
Output: A network $n$
foreach route $r$ of $n.routes$ do
    // Loop through the allocated demand pairs.
    foreach Allocated demand pair $k$ in $r$ do
        if $k$ is not already transhipped then
            var $origin = origin(k)$, var $destination = destination(k)$;
            if path(k) unnecessarily visits turning point then
                // Obtain possible transhipment path candidates.
                var $path\_candidates$ = getTranshipmentCandidates($origin$,$destination$,$n$) ;
                // See Alg.14
                if $count(path\_candidates) > 0$ then
                    $path\_candidates$ = sort($path\_candidates$, path length) ; // sort
                    ascending by path length
                    var $candidatecursor = 1$;
                    while $demand > 0$ && $candidatecursor \leq count(path\_candidates)$ do
                        // The maximum amount we can allocate depends on the
                           remaining capacities of the origin-hub ($RC\_origin\_hub$) and
                           hub-destination ($RC\_hub\_destination$) paths.
                        var $maxAmount$ = min($RC\_origin\_hub$, $RC\_hub\_destination$,
                        $demand$);
                        if $maxAmount > 0$ then
                            if $maxAmount == demand$ then // We can reroute the full
                               demand.
                            Remove allocation $k$ from route $r$. else // We can reroute
                               only part of the demand to this candidate.
                            $r$.Demands($k$) = $r$.Demands($k$) - $maxAmount$;
                            allocateDemand($n.routes(path\_candidates(candidatecursor))$,
                            $origin$, $destination$, $maxAmount$);
                            $demand = demand - maxAmount$;
                        end
                        $candidatecursor = candidatecursor + 1$;
                    end
                end
            end
        end
        $k = k + 1$;
    end
end
return $n$;
```

Algorithm 13: Transhipment operator: allocate remaining OD demand

```
Input: A network $n$
Output: A network $n$
for $i \leftarrow 1$ to $count(n.OD\_matrix)$ do
    for $j \leftarrow 1$ to $count(n.OD\_matrix)$ do
        $demand = n.OD\_matrix(i,j)$;
        if $demand > 0$ then
            // Obtain possible transhipment path candidates.
            var $path\_candidates$ = getTranshipmentCandidates($i$,$j$,$n$) ;        // See Alg. 14
            if $count(path\_candidates) > 0$ then
                $path\_candidates$ = sort($path\_candidates$, path length) ; // sort ascending by
                path length
                var $candidatecursor = 1$;
                while $demand > 0$ && $candidatecursor \leq count(path\_candidates)$ do
                    // The maximum amount we can allocate depends on the remaining
                       capacities of the origin-hub ($RC\_origin\_hub$) and
                       hub-destination ($RC\_hub\_destination$) paths.
                    var $maxAmount$ = min($RC\_origin\_hub$, $RC\_hub\_destination$, $demand$);
                    if $maxAmount > 0$ then
                        allocateDemand($n.routes(path\_candidates(candidatecursor))$, $i$, $j$,
                        $maxAmount$);
                        $demand = demand - maxAmount$;
                    end
                    $candidatecursor = candidatecursor + 1$;
                end
            end
            $n.OD\_matrix(i,j) = demand$;
        end
    end
end
return $n$;
```

Algorithm 14: Transhipment operator: get transhipment path candidates

```
Input: An origin origin, a destination destination, a network n
Output: A list of transhipment path candidates path_candidates
1  var interval = origin:destination;
   // Loop trough the interval origin-destination, these ports are hub candidates.
2  for $m = 2$ to count(interval)-1 do
3      var hub = interval(m);
4      var possible_origins = []; var possible_destinations = [];
5      foreach route r of n.routes do
6          if r.String contains hub then
7              if r.String contains origin then
8                  Find origin and hub position in r.String. Add shortest path between origin
                   and hub to possible_origins.
9              end
10             if r.String contains destination then
11                 Find hub and destination position in r.String. Add shortest path between hub
                   and destination to possible_destinations.
12             end
13         end
14     end
15     if count(possible_origins) > 0 && count(possible_destinations) > 0 then
16         foreach origin-hub path orighub of possible_origins do
17             foreach hub-destination path hubdest of possible_destinations do
18                 if orighub(route)  = hubdest(route) then
19                     path_candidates = [path_candidates; orighub, hubdest];
20                 end
21             end
22         end
23     end
24 end
25 return path_candidates;
```

Algorithm 15: Single-hub single-move (SHSM) transhipment operator

```
Input: A network n
Output: A network n
1  Analogous to the transhipment operator presented in Alg. 11, except for the way transhipment
   path candidates are obtained. The 'getTanshipmentCandidates(origin, destination, n)' function,
   used in line 6 of Alg. 12 and line 5 of Alg. 13, is implemented somewhat different from the
   multi-hub single-move implementation in Alg. 14. Instead of looping through the
   origin-destination interval (line 2 in Alg. 14), where each iteration considers a different hub port
   from the specified interval, we only allow one fixed hub port. Therefore, the previous for-loop is
   removed in the single-hub single-move variant.
```

Algorithm 16: Multi-hub multi-move (MHMM) transhipment operator

```
1 Analogous to the transhipment operator presented in Alg. 11, except for the way transhipment
  path candidates are obtained. The ‘getTanshipmentCandidates(origin, destination, n)’ function,
  used in line 6 of Alg. 12 and line 5 of Alg. 13, is implemented differently, of which the multi-hub
  multi-move variant is specified in Alg. 17. Furthermore, the maximum amount of demand that
  can be (re)allocated (line 11 in Alg. 12 and line 10 in Alg. 13) may also depend on the minimum
  remaining capacity over multiple transhipment paths.
```

Algorithm 17: MHMM transhipment operator: get transhipment path candidates

```
   Input: An origin origin, a destination destination, a network n
   Output: A list of transhipment path candidates path_candidates
1  var interval = origin:destination;
   // Walk trough the interval origin-destination, building the path_candidates as we
      go.
2  for $m = 1$ to count(interval) do
3      var routes_with_m = the routenumbers of routes that stop at port $m$.
4      if $m == 1$ then
           // We have no existing path candidates.
           // This is the origin port, store all routenumbers from routes_with_m in
              variable path_candidates, together with $m$ and the portindex in the
              routes.
5      else
           // We already have path candidates, extend them if possible.
6          First, for each route in routes_with_m, we search for path candidates in path_candidates
           that are currently on that route. Whenever we find a match, we create a copy of the
           path candidate and extend it with the portindex of port interval($m$).
7          Second, if there are 2 or more routes in routes_with_m, we create additional path
           candidates by transhipping copies of existing candidates, that are currently on one of
           the routes, to another route.
8          Every time we extend a candidate we make sure the number of transhipment moves of
           the candidate does not cross the limit of 5.
9          if interval($m$) == destination then
               // We are at the destination, finalize path_candidates.
10             Delete all candidates that do not end at the destination port or that have no
               transhipment moves. Determine each candidate’s path length and minimum
               remaining capacity. Throw away the candidates that have no remaining capacity.
11         end
12     end
13 end
14 return path_candidates;
```

Appendix B

Maersk Asia-Europe data set

This appendix chapter contains all data of the Maersk Asia-Europe (MAE) data set, as well as the data that was used to construct the set. In Table B.1, the original Asia-Europe service network of Maersk is given, that was used to create the data set. Table B.2 shows the vessels operating on these services. An overview of the ports in the MAE data set, with the port names, port codes, country, and throughput, is given in Table B.3. In this case, the throughput corresponds to the general troughput, i.e., not specifically related to Asia-Europe demand. The throughput is used to determine the port's importance with regard to the other ports. Table B.4 specifies the fleet of the data set, with the capacity of each vessel. In the table on page 190, we present the full demand matrix of the MAE data set. The table on page 191 shows the distance matrix. Finally, Table B.7 specifies the natural order of the ports in the data set. Recall that we assumed that the ports lay in a natural order, such that the complex traveling salesman problem is avoided. Although the demand and distance matrix are displayed in alphabetical order, in the core data set file these matrices are shifted to fit the natural order. A map of the ports in the data set is given in Figure B.1.

Table B.1: Services in the original Maersk Asia-Europe network

AE1/AE10	**AE10/AE1**	**AE2**	**AE3**	**AE6**
Yokohama	Shenzhen Yantian	Busan	Dalian	Yokohama
Hong Kong	Hong Kong	Xingang	Xingang	Nagoya
Shenzhen Yantian	Tanjung Pelepas	Dalian	Busan	Shanghai
Tanjung Pelepas	Le Havre	Qingdao	Shanghai	Ningbo
Felixstowe	Zeebrugge	Kwangyang	Ningbo	Xiamen
Rotterdam	Hamburg	Shanghai	Taipei	Hong Kong
Hamburg	Gdansk	Bremerhaven	Shenzhen Chiwan	Shenzhen Yantian
Bremerhaven	Gothenburg	Hamburg	Shenzhen Yantian	Tanjung Pelepas
Tangiers	Aarhus	Rotterdam	Tanjung Pelepas	Jeddah
Jeddah	Bremerhaven	Felixstowe	Port Klang	Barcelona
Jebel Ali	Rotterdam	Antwerp	Port Said	Valencia
Shenzhen Da Chan Bay	Singapore	Tanjung Pelepas	Damietta	Algeciras
Ningbo	Hong Kong	Busan	Izmit	Tangiers
Shanghai	Kobe		Istanbul Ambarli	Tanjung Pelepas
Kaohsiung	Nagoya		Constantza	Vung Tau
	Shimizu		Ilyichevsk	Shenzhen Yantian
	Yokohama		Odessa	Hong Kong
			Damietta	Los Angeles
			Port Said	Yokohama
			Port Klang	
			Tanjung Pelepas	
			Dalian	

AE7	**AE9**	**AE11**	**AE12**
Shanghai	Laem Chabang	Qingdao	Shanghai
Ningbo	Tanjung Pelepas	Shanghai	Busan
Xiamen	Port Klang	Fuzhou	Hong Kong
Hong Kong	Colombo	Hong Kong	Shenzhen Chiwan
Shenzhen Yantian	Zeebrugge	Shenzhen Chiwan	Tanjung Pelepas
Algeciras	Felixstowe	Shenzhen Yantian	Port Klang
Tangiers	Bremerhaven	Tanjung Pelepas	Port Said
Rotterdam	Rotterdam	Port Klang	Piraeus
Felixstowe	Le Havre	Salalah	Koper
Bremerhaven	Tangiers	Port Said	Rijeka
Malaga	Salalah	Gioia Tauro	Trieste
Shenzhen Yantian	Colombo	Genoa	Damietta
Hong Kong	Port Klang	Fos	Port Said
Shanghai	Singapore	Gioia Tauro	Jeddah
	Laem Chabang	Damietta	Port Klang
		Port Said	Singapore
		Salalah	Shanghai
		Port Klang	
		Singapore	
		Liangyungang	
		Qingdao	

Table B.2: Vessels operating in the original Maersk Asia-Europe network

AE1/AE10	**AE10/AE1**	**AE2**	**AE3**	**AE6**
Sofie Maersk	A.P. Moller	Maersk Seville	Maersk Kinloss	Mathilde Maersk
Albert Maersk	Skagen Maersk	Maersk Saigon	CMA CGM Debussy	Maersk Antares
Carsten Maersk	Sally Maersk	Adrian Maersk	Maersk Kuantan	Gunvor Maersk
Maersk Singapore	Arnold Maersk	Maersk Salina	Maersk Kowloon	Mette Maersk
Clementine Maersk	Svendborg Maersk	Maersk Savannah	CMA CGM Corneille	Marit Maersk
Maersk Seoul	Svend Maersk	Anna Maersk	Maersk Kelso	Gerd Maersk
Maersk Taurus	Columbine Maersk	Arthur Maersk	CMA CGM Musset	Maersk Altair
Sine Maersk	Maersk Tukang	Maersk Stepnica	Maersk Kwangyang	Gudrun Maersk
Axel Maersk	Clifford Maersk	Maersk Semarang	CMA CGM Bizet	Marchen Maersk
Cornelia Maersk	Maersk Salalah	Maersk Stralsund	Maersk Kensington	Maren Maersk
	Maersk Stockholm		CMA CGM Baudelaire	Georg Maersk
				Grete Maersk
				Maersk Alfirk
				Margrethe Maersk

AE7	**AE9**	**AE11**	**AE12**
Eugen Maersk	Maersk Sembawang	Charlotte Maersk	Maersk Kyrenia
Elly Maersk	Maersk Sebarok	Maersk Surabaya	Safmarine Komati
Evelyn Maersk	Maersk Serangoon	Maersk Santana	CMA CGM Berlioz
Edith Maersk	SL New York	CMA CGM Faust	Safmarine Kariba
Estelle Maersk	Maersk Seletar	Soroe Maersk	CMA CGM Balzac
Maersk Algol	Maersk Kendal	Susan Maersk	Maersk Karachi
Ebba Maersk	Maersk Sentosa	Caroline Maersk	CMA CGM Ravel
Eleonora Maersk	Maersk Semakau	Cornelius Maersk	CMA CGM Flaubert
Emma Maersk	Maersk Senang	Chastine Maersk	CMA CGM Voltaire
Gjertrud Maersk			

Table B.3: Ports in the MAE data set

Port name	**Code**	**Country**	**Region**	**Visits***	**Troughput (TEU)**
Aarhus	DK AAR	Denmark	Europe	1	683,000
Algeciras	ES ALG	Spain	Europe	2	3,042,759
Antwerp	BE ANR	Belgium	Europe	1	7,309,639
Barcelona	ES BCN	Spain	Europe	1	1,800,213
Bremerhaven	DE BRV	Germany	Europe	5	4,535,842
Busan	KR PUS	South Korea	Asia	3	11,954,861
Colombo	LK CMB	Sri Lanka	Asia	2	3,464,297
Constantza	RO CND	Romania	Europe	1	594,299
Dalian	CN DLC	China	Asia	2	4,552,000
Damietta	EG DAM	Egypt	Middle East	4	1,109,236
Felixstowe	GB FXT	United Kingdom	Europe	4	3,100,000
Fos	FR FOS	France	Europe	1	882,580
Fuzhou	CN FOC	China	Asia	1	1,176,600
Gdansk	PL GDN	Poland	Europe	1	240,623
Genoa	IT GOA	Italy	Europe	1	1,533,627
Gioia Tauro	IT GIT	Italy	Europe	2	2,800,000
Gothenburg	SE GOT	Sweden	Europe	1	817,617
Hamburg	DE HAM	Germany	Europe	3	7,010,000
Hong Kong	CN HOK	China	Asia	9	20,983,000
Ilyichevsk	UA ILK	Ukraine	Europe	1	95,119
Istanbul Ambarli	TR AMB	Turkey	Europe	1	1,835,986
Izmit	TR IZT	Turkey	Europe	1	156,321
Jebel Ali	AE JEA	Dubai	Middle East	1	11,124,082
Jeddah	SA JED	Saudi Arabia	Middle East	3	3,091,312
Kaohsiung	TW KHH	Taiwan	Asia	1	8,581,273
Kobe	JP UKB	Japan	Asia	1	2,247,024
Koper	SI KOP	Slovenia	Europe	1	343,165
Kwangyang	KR KAN	South Korea	Asia	1	1,810,438
Laem Chabang	TH LCH	Thailand	Asia	1	4,621,635
Le Havre	FR LEH	France	Europe	2	2,200,000
Liangyungang	CN LYG	China	Asia	1	3,020,800
Malaga	ES AGP	Spain	Europe	1	289,871
Nagoya	JP NGO	Japan	Asia	2	2,112,743
Ningbo	CN NGB	China	Asia	4	10,502,800
Odessa	UA ODS	Ukraine	Europe	1	123,260
Piraeus	GR PIR	Greece	Europe	1	1,403,408
Port Klang	MY PKL	Malaysia	Asia	8	7,309,779
Port Said	EG PSD	Egypt	Middle East	6	3,470,000
Qingdao	CN TAO	China	Asia	2	10,260,000
Rijeka	HR RJK	Croatia	Europe	1	145,000
Rotterdam	NL RTM	Netherlands	Europe	5	9,743,290
Salalah	OM SLL	Oman	Middle East	3	3,490,000
Shanghai	CN SHA	China	Asia	7	25,002,000
Shenzhen Chiwan	CN CWN	China	Asia	3	4,562,525
Shenzhen Da Chan Bay	CN SAD	China	Asia	1	1,520,842
Shenzhen Yantian	CN YTN	China	Asia	8	12,166,733
Shimizu	JP SMZ	Japan	Asia	1	500,000
Singapore	SG SIN	Singapore	Asia	4	25,866,400
Taipei	TW TAP	Taiwan	Asia	1	1,000,000
Tangiers	MA TNG	Marocco	Europe	4	1,000,000
Tanjung Pelepas	MY TPP	Malaysia	Asia	10	6,000,000
Trieste	IT TRS	Italy	Europe	1	276,957
Valencia	ES VLC	Spain	Europe	1	3,653,890
Vung Tau	VN SGN	Vietnam	Asia	1	1,849,746
Xiamen	CN XMN	China	Asia	2	4,680,355
Xingang	CN TXG	China	Asia	2	8,700,000
Yokohama	JP YOK	Japan	Asia	2	2,798,002
Zeebrugge	BE ZEE	Belgium	Europe	2	2,328,198

(*) Corresponds to the number of visits per port in the original service network.

Table B.4: Vessels in the MAE data set

Vessel name	Capacity (TEU)
A.P. Moller	8160
Adrian Maersk	8272
Albert Maersk	8272
Anna Maersk	8272
Arnold Maersk	8272
Arthur Maersk	8272
Axel Maersk	8272
Caroline Maersk	8160
Carsten Maersk	8160
Charlotte Maersk	8194
Chastine Maersk	8160
Clementine Maersk	8648
Clifford Maersk	8160
CMA CGM Balzac	6251
CMA CGM Baudelaire	6251
CMA CGM Berlioz	6627
CMA CGM Bizet	6627
CMA CGM Corneille	6500
CMA CGM Debussy	6627
CMA CGM Faust	8204
CMA CGM Flaubert	6638
CMA CGM Musset	6540
CMA CGM Ravel	6712
CMA CGM Voltaire	6456
Columbine Maersk	8648
Cornelia Maersk	8650
Cornelius Maersk	8160
Ebba Maersk	14770
Edith Maersk	14770
Eleonora Maersk	14770
Elly Maersk	14770
Emma Maersk	14770
Estelle Maersk	14770
Eugen Maersk	14770
Evelyn Maersk	14770
Georg Maersk	9074
Gerd Maersk	9074
Gjertrud Maersk	9074
Grete Maersk	9074
Gudrun Maersk	9074
Gunvor Maersk	9074
Maersk Alfirk	9200
Maersk Algol	9200
Maersk Altair	9200
Maersk Antares	9200
Maersk Karachi	6930
Maersk Kelso	6500
Maersk Kendal	6500
Maersk Kensington	6500
Maersk Kinloss	6500
Maersk Kowloon	6500
Maersk Kuantan	6500
Maersk Kwangyang	6500
Maersk Kyrenia	6978
Maersk Saigon	8450
Maersk Salalah	8600
Maersk Salina	8600
Maersk Santana	8478
Maersk Savannah	8600
Maersk Sebarok	6478
Maersk Seletar	6478
Maersk Semakau	6478
Maersk Semarang	8400
Maersk Sembawang	6478
Maersk Senang	6478
Maersk Sentosa	6478
Maersk Seoul	8450
Maersk Serangoon	6478
Maersk Seville	8478
Maersk Singapore	8478
Maersk Stepnica	8600
Maersk Stockholm	8600
Maersk Stralsund	8450
Maersk Surabaya	8400
Maersk Taurus	8400
Maersk Tukang	8400
Margrethe Maersk	9038
Marit Maersk	9038
Mathilde Maersk	9038
Mette Maersk	9038
Safmarine Kariba	6500
Safmarine Komati	6500
Sally Maersk	8160
Sine Maersk	8160
Skagen Maersk	8160
SL New York	6420
Sofie Maersk	8160
Soroe Maersk	8160
Susan Maersk	8160
Svend Maersk	8160
Svendborg Maersk	8160

Demand Matrix

Destination	Aarhus (DK AAR)	Algeciras (ES ALG)	Antwerp (BE ANR)	Barcelona (ES BCN)	Bremerhaven (DE BRV)	Busan (KR PUS)	Colombo (LK CMB)	Constantza (RO CND)	Dalian (CN DLC)	Damietta (EG DAM)	Felixstowe (GB FXT)	Fos (FR FOS)	Fuzhou (CN FOC)	Gdansk (PL GDN)	Genoa (IT GOA)	Gioia Tauro (IT GIT)	Gothenburg (SE GOT)	Hamburg (DE HAM)	Hong Kong (CN HOK)	Ilyichevsk (UA ILK)	Istanbul Ambarli (TR AMB)	Izmit (TR IZT)	Jebel Ali Terminal (AE JEA)	Jeddah (SA JED)	Kaohsiung (TW KHH)	Kobe (JP UKB)	Koper (SI KOP)	Kwangyang (KR KAN)	Laem Chabang (TH LCH)	Le Havre (FR LEH)	Liangyungang (CN LYG)	Malaga (ES AGP)	Nagoya (JP NGO)	Ningbo (CN NGB)	Odessa (UA ODS)	Piraeus (GR PIR)	Port Klang (MY PKL)	Port Said (EG PSD)	Qingdao (CN TAO)	Rijeka (HR RJK)	Rotterdam (NL RTM)	Salalah (OM SLL)	Shanghai (CN SHA)	Shenzhen Chiwan (CN CWN)	Shenzhen Da Chan Bay (CN SAD)	Shenzhen Yantian (CN YTN)	Shimizu (JP SMZ)	Singapore (SG SIN)	Taipei (TW TAP)	Tangiers (MA TNG)	Tanjung Pelepas (MY TPP)	Trieste (IT TRS)	Valencia (ES VLC)	Vung Tau (VN SGN)	Xiamen (CN XMN)	Xingang (CN TXG)	Yokohama (JP YOK)	Zeebrugge (BE ZEE)
Origin																																																										
Aarhus (DK AAR)	0	0	0	0	0	987	286	0	376	92	0	0	97	0	0	0	0	0	1732	0	0	0	918	255	708	186	0	149	382	0	249	0	174	867	0	0	604	286	847	0	0	288	2064	377	126	1005	41	2136	83	0	495	0	0	153	386	718	231	0
Algeciras (ES ALG)	0	0	0	0	0	4397	1274	0	1674	408	0	0	433	0	0	0	0	0	7718	0	0	0	4092	1137	3156	826	0	666	1700	0	1111	0	777	3863	0	0	2689	1276	3774	0	0	1284	9196	1678	559	4475	184	9514	368	0	2207	0	0	680	1721	3200	1029	0
Antwerp (BE ANR)	0	0	0	0	0	10563	3061	0	4022	980	0	0	1040	0	0	0	0	0	18540	0	0	0	9829	2731	7582	1985	0	1600	4084	0	2669	0	1867	9280	0	0	6459	3066	9066	0	0	3084	22091	4031	1344	10750	442	22855	884	0	5302	0	0	1634	4135	7687	2472	0
Barcelona (ES BCN)	0	0	0	0	0	2601	754	0	991	241	0	0	256	0	0	0	0	0	4566	0	0	0	2421	673	1867	489	0	394	1006	0	657	0	460	2285	0	0	1591	755	2233	0	0	759	5441	993	331	2648	109	5629	218	0	1306	0	0	403	1018	1893	609	0
Bremerhaven (DE BRV)	0	0	0	0	0	6555	1899	0	2496	608	0	0	645	0	0	0	0	0	11505	0	0	0	6099	1695	4705	1232	0	993	2534	0	1656	0	1158	5759	0	0	4008	1903	5625	0	0	1914	13708	2502	834	6671	274	14182	548	0	3290	0	0	1014	2566	4770	1534	0
Busan (KR PUS)	2578	11484	27587	6794	17118	0	0	2243	0	4186	11700	3331	0	908	5788	10567	3086	26456	0	359	6929	590	41983	11667	0	0	1295	0	0	8303	0	1094	0	0	465	5297	0	13096	0	547	36772	13171	0	0	0	0	0	0	0	3774	0	1045	13790	0	0	0	0	8787
Colombo (LK CMB)	747	3328	7994	1969	4961	0	0	650	0	1213	3390	965	0	263	1677	3062	894	7667	0	104	2008	171	12166	3381	0	0	375	0	0	2406	0	317	0	0	135	1535	0	3795	0	159	10656	3817	0	0	0	0	0	0	0	1094	0	303	3996	0	0	0	0	2546
Constantza (RO CND)	0	0	0	0	0	859	249	0	327	80	0	0	85	0	0	0	0	0	1507	0	0	0	799	222	616	161	0	130	332	0	217	0	152	754	0	0	525	249	737	0	0	251	1796	328	109	874	36	1858	72	0	431	0	0	133	336	625	201	0
Dalian (CN DLC)	981	4373	10504	2587	6518	0	0	854	0	1594	4455	1268	0	346	2204	4024	1175	10074	0	137	2638	225	15986	4442	0	0	493	0	0	3161	0	417	0	0	177	2017	0	4986	0	208	14001	5015	0	0	0	0	0	0	0	1437	0	398	5251	0	0	0	0	3346
Damietta (EG DAM)	78	349	838	206	520	1370	397	68	522	0	355	101	135	28	176	321	94	803	2404	11	210	18	0	0	983	257	39	207	530	252	346	33	242	1203	14	161	838	0	1176	17	1116	0	2865	523	174	1394	57	2964	115	115	687	32	419	212	536	997	321	267
Felixstowe (GB FXT)	0	0	0	0	0	4480	1298	0	1706	416	0	0	441	0	0	0	0	0	7863	0	0	0	4168	1158	3216	842	0	678	1732	0	1132	0	792	3936	0	0	2739	1300	3845	0	0	1308	9369	1710	570	4559	187	9693	375	0	2248	0	0	693	1754	3260	1048	0
Fos (FR FOS)	0	0	0	0	0	1275	370	0	486	118	0	0	126	0	0	0	0	0	2239	0	0	0	1187	330	916	240	0	193	493	0	322	0	225	1121	0	0	780	370	1095	0	0	372	2667	487	162	1298	53	2760	107	0	640	0	0	197	499	928	299	0
Fuzhou (CN FOC)	254	1130	2715	669	1685	0	0	221	0	412	1151	328	0	89	570	1040	304	2604	0	35	682	58	4132	1148	0	0	127	0	0	817	0	108	0	0	46	521	0	1289	0	54	3619	1296	0	0	0	0	0	0	0	371	0	103	1357	0	0	0	0	865
Gdansk (PL GDN)	0	0	0	0	0	348	101	0	132	32	0	0	34	0	0	0	0	0	610	0	0	0	324	90	250	65	0	53	134	0	88	0	61	305	0	0	213	101	298	0	0	102	727	133	44	354	15	752	29	0	175	0	0	54	136	253	81	0
Genoa (IT GOA)	0	0	0	0	0	2216	642	0	844	206	0	0	218	0	0	0	0	0	3890	0	0	0	2062	573	1591	417	0	336	857	0	560	0	392	1947	0	0	1355	643	1902	0	0	647	4635	846	282	2256	93	4795	185	0	1112	0	0	343	868	1613	519	0
Gioia Tauro (IT GIT)	0	0	0	0	0	4046	1173	0	1541	375	0	0	398	0	0	0	0	0	7102	0	0	0	3765	1046	2904	761	0	613	1564	0	1022	0	715	3555	0	0	2474	1174	3473	0	0	1181	8462	1544	515	4118	169	8755	338	0	2031	0	0	626	1584	2945	947	0
Gothenburg (SE GOT)	0	0	0	0	0	1182	342	0	450	110	0	0	116	0	0	0	0	0	2074	0	0	0	1099	306	848	222	0	179	457	0	299	0	209	1038	0	0	722	343	1014	0	0	345	2471	451	150	1202	49	2556	99	0	593	0	0	183	463	860	277	0
Hamburg (DE HAM)	0	0	0	0	0	10130	2936	0	3857	940	0	0	997	0	0	0	0	0	17780	0	0	0	9426	2619	7271	1904	0	1534	3916	0	2560	0	1790	8900	0	0	6194	2940	8694	0	0	2957	21186	3866	1289	10310	424	21918	847	0	5084	0	0	1567	3966	7372	2371	0
Hong Kong (CN HOK)	4524	20156	48420	11925	30046	0	0	3937	0	7348	20535	5846	0	1594	10159	18548	5416	46435	0	630	12162	1035	73688	20477	0	0	2273	0	0	14573	0	1920	0	0	816	9296	0	22986	0	961	64541	23118	0	0	0	0	0	0	0	6624	0	1835	24204	0	0	0	0	15422
Ilyichevsk (UA ILK)	0	0	0	0	0	137	40	0	52	13	0	0	14	0	0	0	0	0	241	0	0	0	128	36	99	26	0	21	53	0	35	0	24	121	0	0	84	40	118	0	0	40	287	52	17	140	6	297	11	0	69	0	0	21	54	100	32	0
Istanbul Ambarli (TR AMB)	0	0	0	0	0	2653	769	0	1010	246	0	0	261	0	0	0	0	0	4657	0	0	0	2469	686	1904	499	0	402	1026	0	670	0	469	2331	0	0	1622	770	2277	0	0	775	5549	1013	338	2700	111	5741	222	0	1332	0	0	411	1039	1931	621	0
Izmit (TR IZT)	0	0	0	0	0	226	65	0	86	21	0	0	22	0	0	0	0	0	396	0	0	0	210	58	162	42	0	34	87	0	57	0	40	198	0	0	138	66	194	0	0	66	472	86	29	230	9	489	19	0	113	0	0	35	88	164	53	0
Jebel Ali Terminal (AE JEA)	785	3496	8400	2069	5212	13737	3981	683	5231	0	3562	1014	1352	277	1762	3217	940	8055	24112	109	2110	180	0	0	9861	2582	394	2080	5311	2528	3471	333	2428	12069	142	1613	8400	0	11790	167	11196	0	28730	5243	1748	13981	575	29723	1149	1149	6895	318	4199	2126	5378	9997	3215	2675
Jeddah (SA JED)	218	972	2334	575	1448	3818	1106	190	1454	0	990	282	376	77	490	894	261	2239	6700	30	586	50	0	0	2740	718	110	578	1476	703	965	93	675	3354	39	448	2334	0	3276	46	3111	0	7984	1457	486	3885	160	8260	319	319	1916	88	1167	591	1495	2778	893	743
Kaohsiung (TW KHH)	1850	8243	19802	4877	12288	0	0	1610	0	3005	8398	2391	0	652	4155	7585	2215	18990	0	258	4974	423	30136	8374	0	0	930	0	0	5960	0	785	0	0	334	3802	0	9400	0	393	26395	9455	0	0	0	0	0	0	0	2709	0	750	9899	0	0	0	0	6307
Kobe (JP UKB)	484	2158	5185	1277	3218	0	0	422	0	787	2199	626	0	171	1088	1986	580	4973	0	67	1302	111	7891	2193	0	0	243	0	0	1561	0	206	0	0	87	996	0	2462	0	103	6912	2476	0	0	0	0	0	0	0	709	0	196	2592	0	0	0	0	1652
Koper (SI KOP)	0	0	0	0	0	496	144	0	189	46	0	0	49	0	0	0	0	0	870	0	0	0	461	128	356	93	0	75	192	0	125	0	88	436	0	0	303	144	426	0	0	145	1037	189	63	505	21	1073	41	0	249	0	0	77	194	361	116	0
Kwangyang (KR KAN)	390	1739	4178	1029	2592	0	0	340	0	634	1772	504	0	138	877	1600	467	4006	0	54	1049	89	6358	1767	0	0	196	0	0	1257	0	166	0	0	70	802	0	1983	0	83	5569	1995	0	0	0	0	0	0	0	572	0	158	2088	0	0	0	0	1331
Laem Chabang (TH LCH)	997	4439	10665	2627	6618	0	0	867	0	1618	4523	1288	0	351	2238	4085	1193	10228	0	139	2679	228	16230	4510	0	0	501	0	0	3210	0	423	0	0	180	2048	0	5063	0	212	14216	5092	0	0	0	0	0	0	0	1459	0	404	5331	0	0	0	0	3397
Le Havre (FR LEH)	0	0	0	0	0	3179	921	0	1211	295	0	0	313	0	0	0	0	0	5580	0	0	0	2958	822	2282	598	0	481	1229	0	803	0	562	2793	0	0	1944	923	2728	0	0	928	6649	1213	404	3236	133	6879	266	0	1596	0	0	492	1245	2314	744	0
Liangyungang (CN LYG)	651	2902	6971	1717	4326	0	0	567	0	1058	2956	842	0	229	1463	2670	780	6685	0	91	1751	149	10608	2948	0	0	327	0	0	2098	0	276	0	0	118	1338	0	3309	0	138	9292	3328	0	0	0	0	0	0	0	954	0	264	3485	0	0	0	0	2220
Malaga (ES AGP)	0	0	0	0	0	419	121	0	160	39	0	0	41	0	0	0	0	0	735	0	0	0	390	108	301	79	0	63	162	0	106	0	74	368	0	0	256	122	360	0	0	122	876	160	53	426	18	906	35	0	210	0	0	65	164	305	98	0
Nagoya (JP NGO)	456	2029	4875	1201	3025	0	0	396	0	740	2068	589	0	160	1023	1868	545	4676	0	63	1225	104	7419	2062	0	0	229	0	0	1467	0	193	0	0	82	936	0	2314	0	97	6499	2328	0	0	0	0	0	0	0	667	0	185	2437	0	0	0	0	1553
Ningbo (CN NGB)	2265	10089	24236	5969	15039	0	0	1970	0	3678	10279	2926	0	798	5085	9284	2711	23243	0	315	6087	518	36884	10250	0	0	1138	0	0	7294	0	961	0	0	409	4653	0	11505	0	481	32305	11572	0	0	0	0	0	0	0	3316	0	918	12115	0	0	0	0	7719
Odessa (UA ODS)	0	0	0	0	0	178	52	0	68	17	0	0	18	0	0	0	0	0	313	0	0	0	166	46	128	33	0	27	69	0	45	0	31	156	0	0	109	52	153	0	0	52	373	68	23	181	7	385	15	0	89	0	0	28	70	130	42	0
Piraeus (GR PIR)	0	0	0	0	0	2028	588	0	772	188	0	0	200	0	0	0	0	0	3560	0	0	0	1887	524	1456	381	0	307	784	0	512	0	358	1782	0	0	1240	589	1741	0	0	592	4241	774	258	2064	85	4388	170	0	1018	0	0	314	794	1476	475	0
Port Klang (MY PKL)	1576	7022	16868	4154	10467	0	0	1371	0	2560	7154	2037	0	555	3539	6461	1887	16177	0	220	4237	361	25670	7134	0	0	792	0	0	5077	0	669	0	0	284	3239	0	8007	0	335	22484	8054	0	0	0	0	0	0	0	2308	0	639	8432	0	0	0	0	5373
Port Said (EG PSD)	245	1091	2620	645	1626	4285	1242	213	1632	0	1111	316	422	86	550	1004	293	2513	7521	34	658	56	0	0	3076	805	123	649	1657	789	1083	104	757	3765	44	503	2620	0	3678	52	3492	0	8962	1635	545	4361	179	9272	358	358	2151	99	1310	663	1678	3118	1003	835
Qingdao (CN TAO)	2212	9855	23676	5831	14692	0	0	1925	0	3593	10041	2859	0	779	4967	9069	2648	22705	0	308	5947	506	36031	10013	0	0	1112	0	0	7126	0	939	0	0	399	4546	0	11239	0	470	31558	11304	0	0	0	0	0	0	0	3239	0	897	11835	0	0	0	0	7541
Rijeka (HR RJK)	0	0	0	0	0	210	61	0	80	19	0	0	21	0	0	0	0	0	368	0	0	0	195	54	150	39	0	32	81	0	53	0	37	184	0	0	128	61	180	0	0	61	438	80	27	213	9	453	18	0	105	0	0	32	82	153	49	0
Rotterdam (NL RTM)	0	0	0	0	0	14080	4080	0	5361	1306	0	0	1386	0	0	0	0	0	24713	0	0	0	13102	3641	10107	2646	0	2132	5443	0	3558	0	2488	12370	0	0	8609	4087	12084	0	0	4110	29446	5374	1791	14330	589	30465	1178	0	7067	0	0	2179	5512	10247	3295	0
Salalah (OM SLL)	246	1097	2635	649	1635	4310	1249	214	1641	0	1118	318	424	87	553	1009	295	2527	7565	34	662	56	0	0	3094	810	124	653	1666	793	1089	105	762	3786	44	506	2635	0	3699	52	3513	0	9014	1645	548	4386	180	9325	361	361	2163	100	1317	667	1687	3136	1009	839
Shanghai (CN SHA)	5391	24016	57694	14209	35801	0	0	4691	0	8755	24468	6966	0	1899	12105	22100	6453	55329	0	751	14491	1234	87801	24399	0	0	2709	0	0	17364	0	2288	0	0	973	11077	0	27388	0	1144	76903	27546	0	0	0	0	0	0	0	7893	0	2186	28840	0	0	0	0	18376
Shenzhen Chiwan (CN CWN)	984	4383	10528	2593	6533	0	0	856	0	1598	4465	1271	0	347	2209	4033	1178	10097	0	137	2644	225	16023	4453	0	0	494	0	0	3169	0	418	0	0	178	2021	0	4998	0	209	14034	5027	0	0	0	0	0	0	0	1440	0	399	5263	0	0	0	0	3353
Shenzhen Da Chan Bay (CN SAD)	328	1461	3509	864	2178	0	0	285	0	533	1488	424	0	116	736	1344	393	3366	0	46	881	75	5341	1484	0	0	165	0	0	1056	0	139	0	0	59	674	0	1666	0	70	4678	1676	0	0	0	0	0	0	0	480	0	133	1754	0	0	0	0	1118
Shenzhen Yantian (CN YTN)	2623	11687	28076	6914	17422	0	0	2283	0	4260	11907	3390	0	924	5891	10755	3140	26925	0	365	7052	600	42727	11874	0	0	1318	0	0	8450	0	1113	0	0	473	5390	0	13328	0	557	37423	13405	0	0	0	0	0	0	0	3841	0	1064	14034	0	0	0	0	8942
Shimizu (JP SMZ)	108	480	1154	284	716	0	0	94	0	175	489	139	0	38	242	442	129	1107	0	15	290	25	1756	488	0	0	54	0	0	347	0	46	0	0	19	222	0	548	0	23	1538	551	0	0	0	0	0	0	0	158	0	44	577	0	0	0	0	368
Singapore (SG SIN)	5577	24847	59689	14700	37039	0	0	4853	0	9058	25314	7207	0	1965	12523	22864	6676	57242	0	777	14992	1276	90837	25243	0	0	2802	0	0	17965	0	2367	0	0	1007	11460	0	28335	0	1184	79562	28499	0	0	0	0	0	0	0	8166	0	2262	29837	0	0	0	0	19012
Taipei (TW TAP)	216	961	2308	568	1432	0	0	188	0	350	979	279	0	76	484	884	258	2213	0	30	580	49	3512	976	0	0	108	0	0	695	0	92	0	0	39	443	0	1095	0	46	3076	1102	0	0	0	0	0	0	0	316	0	87	1154	0	0	0	0	735
Tangiers (MA TNG)	0	0	0	0	0	1445	419	0	550	134	0	0	142	0	0	0	0	0	2536	0	0	0	1345	374	1037	272	0	219	559	0	365	0	255	1270	0	0	884	419	1240	0	0	422	3022	552	184	1471	60	3127	121	0	725	0	0	224	566	1052	338	0
Tanjung Pelepas (MY TPP)	1294	5763	13846	3410	8592	0	0	1126	0	2101	5872	1672	0	456	2905	5304	1549	13278	0	180	3478	296	21071	5855	0	0	650	0	0	4167	0	549	0	0	233	2658	0	6573	0	275	18455	6611	0	0	0	0	0	0	0	1894	0	525	6921	0	0	0	0	4410
Trieste (IT TRS)	0	0	0	0	0	400	116	0	152	37	0	0	39	0	0	0	0	0	703	0	0	0	372	103	287	75	0	61	155	0	101	0	71	352	0	0	245	116	344	0	0	117	837	153	51	407	17	866	33	0	201	0	0	62	157	291	94	0
Valencia (ES VLC)	0	0	0	0	0	5280	1530	0	2011	490	0	0	520	0	0	0	0	0	9268	0	0	0	4913	1365	3790	992	0	800	2041	0	1334	0	933	4639	0	0	3229	1533	4532	0	0	1541	11043	2015	672	5374	221	11425	442	0	2650	0	0	817	2067	3843	1236	0
Vung Tau (VN SGN)	399	1777	4268	1051	2649	0	0	347	0	648	1810	515	0	141	896	1635	477	4093	0	56	1072	91	6496	1805	0	0	200	0	0	1285	0	169	0	0	72	820	0	2026	0	85	5690	2038	0	0	0	0	0	0	0	584	0	162	2134	0	0	0	0	1360
Xiamen (CN XMN)	1009	4496	10800	2660	6702	0	0	878	0	1639	4580	1304	0	356	2266	4137	1208	10358	0	141	2713	231	16436	4568	0	0	507	0	0	3251	0	428	0	0	182	2074	0	5127	0	214	14396	5157	0	0	0	0	0	0	0	1478	0	409	5399	0	0	0	0	3440
Xingang (CN TXG)	1876	8357	20076	4944	12458	0	0	1632	0	3047	8514	2424	0	661	4212	7690	2246	19253	0	261	5043	429	30552	8490	0	0	943	0	0	6042	0	796	0	0	339	3854	0	9530	0	398	26760	9585	0	0	0	0	0	0	0	2747	0	761	10035	0	0	0	0	6394
Yokohama (JP YOK)	603	2688	6457	1590	4007	0	0	525	0	980	2738	780	0	213	1355	2473	722	6192	0	84	1622	138	9826	2731	0	0	303	0	0	1943	0	256	0	0	109	1240	0	3065	0	128	8606	3083	0	0	0	0	0	0	0	883	0	245	3228	0	0	0	0	2057
Zeebrugge (BE ZEE)	0	0	0	0	0	3364	975	0	1281	312	0	0	331	0	0	0	0	0	5905	0	0	0	3131	870	2415	632	0	510	1301	0	850	0	595	2956	0	0	2057	977	2887	0	0	982	7036	1284	428	3424	141	7280	281	0	1689	0	0	521	1317	2448	787	0

Distance Matrix

Destination	Aarhus (DK AAR)	Algeciras (ES ALG)	Antwerp (BE ANR)	Barcelona (ES BCN)	Bremerhaven (DE BRV)	Busan (KR PUS)	Colombo (LK CMB)	Constantza (RO CND)	Dalian (CN DLC)	Damietta (EG DAM)	Felixstowe (GB FXT)	Fos (FR FOS)	Fuzhou (CN FOC)	Gdansk (PL GDN)	Genoa (IT GOA)	Gioia Tauro (IT GIT)	Gothenburg (SE GOT)	Hamburg (DE HAM)	Hong Kong (CN HOK)	Ilyichevsk (UA ILK)	Istanbul Ambarli (TR AMB)	Izmit (TR IZT)	Jebel Ali Terminal (AE JEA)	Jeddah (SA JED)	Kaohsiung (TW KHH)	Kobe (JP UKB)	Koper (SI KOP)	Kwangyang (KR KAN)	Laem Chabang (TH LCH)	Le Havre (FR LEH)	Liangyungang (CN LYG)	Malaga (ES AGP)	Nagoya (JP NGO)	Ningbo (CN NGB)	Odessa (UA ODS)	Piraeus (GR PIR)	Port Klang (MY PKL)	Port Said (EG PSD)	Qingdao (CN TAO)	Rijeka (HR RJK)	Rotterdam (NL RTM)	Salalah (OM SLL)	Shanghai (CN SHA)	Shenzhen Chiwan (CN CWN)	Shenzhen Da Chan Bay (CN SAD)	Shenzhen Yantian (CN YTN)	Shimizu (JP SMZ)	Singapore (SG SIN)	Taipei (TW TAP)	Tangiers (MA TNG)	Tanjung Pelepas (MY TPP)	Trieste (IT TRS)	Valencia (ES VLC)	Vung Tau (VN SGN)	Xiamen (CN XMN)	Xingang (CN TXG)	Yokohama (JP YOK)	Zeebrugge (BE ZEE)
Origin																																																										
Aarhus (DK AAR)	0	1892	680	2403	456	11312	7280	3884	11428	3701	606	2578	10634	379	2735	2911	151	485	10269	4017	3673	3720	6709	4522	10430	11500	3532	11212	9640	774	11025	1951	11599	10721	4030	3369	8616	3801	11272	3496	584	5626	11046	10279	10284	10289	11633	8815	10454	1869	8809	3536	2275	9455	10462	11391	11701	627
Algeciras (ES ALG)	1892	0	1370	513	1550	9409	5402	1995	9502	1930	1278	680	8710	2131	847	1022	1789	1618	8336	2131	1792	1643	4767	2625	8538	9606	1642	9358	7754	1155	9286	65	9697	8978	2139	1484	6744	1913	9334	1603	1355	3885	9050	8393	8398	8363	9744	6926	8735	32	6905	1650	382	7569	8564	9646	9805	1294
Antwerp (BE ANR)	680	1370	0	1881	357	10790	6758	3362	10906	3271	141	2029	10112	970	2213	2389	597	405	9747	3495	3166	3198	6187	4000	9908	10978	3010	10733	9118	252	10634	1429	11077	10330	3508	2847	8094	3279	10750	2974	149	5234	10524	9757	9762	9701	11111	8293	10099	1347	8246	3014	1753	8933	9940	10999	11179	87
Barcelona (ES BCN)	2403	513	1881	0	2058	9081	5074	1664	9174	1603	1786	172	8382	2639	350	671	2297	2129	8008	1798	1460	1504	4439	2297	8210	9278	1290	9030	7429	1663	8958	510	9369	8650	1808	1153	6416	1585	9006	1251	1863	3557	8722	8068	8073	8035	9416	6598	8410	585	6577	1301	161	7244	8236	9318	9477	1802
Bremerhaven (DE BRV)	456	1550	357	2058	0	10933	6926	3520	11026	3471	303	2204	10234	686	2372	2547	344	117	9860	3695	3316	3398	6291	4149	10062	11130	3166	10882	9318	431	10810	1629	11221	10502	3664	3009	8268	3437	10858	3127	215	5409	10574	9957	9962	9887	11268	8450	10299	1547	8429	3214	1907	9133	10088	11170	11329	269
Busan (KR PUS)	11312	9409	10790	9081	10933	0	4036	8478	503	7526	10672	9010	752	11525	8910	8427	11183	11038	1137	8626	8274	8335	5909	6795	898	341	8783	72	2569	10549	493	9375	511	495	8622	8086	2677	7499	467	8744	10749	5650	445	1150	1155	1120	560	2483	716	9448	2504	8805	9161	2005	869	654	626	10688
Colombo (LK CMB)	7280	5402	6758	5074	6926	4036	0	4471	4129	3508	6665	5003	3337	7518	4903	4420	7176	7006	2963	4594	4267	4303	1887	2788	3164	4232	4776	3985	2398	6542	3913	5343	4324	3604	4615	4079	1370	3492	3961	4737	6742	1633	3677	3037	3042	2990	4371	1553	3379	5416	1532	4773	5154	2213	3191	4273	4432	6681
Constantza (RO CND)	3884	1995	3362	1664	3520	8478	4471	0	8571	1012	3259	1576	7779	4112	1477	994	3770	3610	7405	160	203	251	3836	1694	7607	8675	1340	8427	6821	3136	8355	1942	8766	8047	170	553	5813	982	8403	1301	3336	2954	8119	7460	7465	7432	8813	5995	7802	2020	5974	1348	1745	6636	7633	8715	8874	3275
Dalian (CN DLC)	11428	9502	10906	9174	11026	503	4129	8571	0	7630	10765	9103	843	11618	9003	8520	11276	11154	1229	8742	8367	8451	6002	6888	1006	810	8876	437	2679	10642	324	9491	980	556	8715	8179	2770	7592	251	8837	10842	5743	477	1266	1271	1212	1030	2576	851	9564	2597	8921	9254	2119	961	188	1095	10781
Damietta (EG DAM)	3701	1930	3271	1603	3471	7526	3508	1012	7630	0	3197	1536	6832	3867	1434	970	3707	3526	6570	1145	793	850	2922	746	6697	7741	1307	7426	5806	3076	7413	1877	7828	7109	1157	607	4854	31	7458	1265	3277	2013	7181	6580	6585	6590	7870	5036	6842	1951	5025	1310	1683	5668	6674	7779	7941	3212
Felixstowe (GB FXT)	606	1278	141	1786	303	10672	6665	3259	10765	3197	0	1932	9962	872	2100	2275	530	360	9588	3421	3044	3124	6019	3877	9790	10858	2894	10610	9044	163	10538	1355	10949	10230	3392	2737	7996	3165	10586	2855	123	5137	10302	9683	9688	9615	10996	8178	10025	1273	8157	2940	1635	8859	9816	10898	11057	83
Fos (FR FOS)	2578	680	2029	172	2204	9010	5003	1576	9103	1536	1932	0	8311	2806	208	583	2464	2285	7937	1707	1373	1422	4368	2226	8139	9207	1202	8959	7340	1830	8887	641	9298	8579	1720	1066	6345	1513	8935	1164	2030	3486	8651	7990	7995	7964	9345	6527	8332	714	6506	1215	331	7166	8165	9247	9406	1969
Fuzhou (CN FOC)	10634	8710	10112	8382	10234	752	3337	7779	843	6832	9962	8311	0	10826	8211	7728	10484	10360	436	7948	7575	7657	5210	6096	235	999	8084	699	1894	9850	626	8697	1097	318	7923	7387	1978	6800	675	8045	10050	4951	390	472	477	420	1146	1784	150	8770	1805	8127	8462	1327	168	987	1211	9989
Gdansk (PL GDN)	379	2131	970	2639	686	11525	7518	4112	11618	3867	872	2806	10826	0	2953	3128	368	775	10441	4307	3897	4010	6872	4730	10643	11711	3747	11463	9930	1015	11391	2241	11802	11083	4245	3590	8849	4018	11439	3708	809	5990	11155	10569	10574	10468	11849	9031	10911	2159	9010	3826	2488	9745	10669	11751	11910	861
Genoa (IT GOA)	2735	847	2213	350	2372	8910	4903	1477	9003	1434	2100	208	8211	2953	0	484	2631	2461	7837	1614	1273	1323	4268	2126	8039	9107	1103	8859	7258	1997	8787	798	9198	8479	1621	966	6245	1413	8835	1064	2197	3386	8551	7897	7902	7864	9245	6427	8239	871	6406	1111	508	7073	8065	9147	9306	2136
Gioia Tauro (IT GIT)	2911	1022	2389	671	2547	8427	4420	994	8520	970	2275	583	7728	3128	484	0	2787	2637	7354	1128	790	837	3785	1643	7556	8624	620	8376	6772	2152	8304	973	8715	7996	1138	483	5762	931	8352	581	2372	2903	8068	7364	7369	7381	8762	5944	7753	1047	5923	625	758	6587	7582	8664	8823	2311
Gothenburg (SE GOT)	151	1789	597	2297	344	11183	7176	3770	11276	3707	530	2464	10484	368	2631	2787	0	402	10100	3934	3556	3637	6531	4389	10302	11370	3406	11122	9557	674	11050	1868	11461	10742	3904	3249	8508	3677	11098	3367	467	5649	10814	10196	10201	10127	11508	8690	10538	1786	8669	3453	2147	9372	10328	11410	11569	519
Hamburg (DE HAM)	485	1618	405	2129	117	11038	7006	3610	11154	3526	360	2285	10360	775	2461	2637	402	0	9995	3743	3414	3446	6435	4248	10156	11226	3258	10989	9366	500	10889	1677	11325	10585	3756	3095	8342	3527	10998	3222	305	5490	10772	10005	10010	9956	11359	8541	10347	1595	8502	3262	2001	9181	10188	11255	11427	347
Hong Kong (CN HOK)	10269	8336	9747	8008	9860	1137	2963	7405	1229	6570	9588	7937	436	10441	7837	7354	10100	9995	0	7583	7201	7292	4836	5722	354	1369	7710	1084	1489	9476	1012	8332	1467	704	7549	7013	1604	6426	1061	7671	9676	4577	776	10	15	46	1516	1410	471	8405	1431	7762	8088	927	287	1372	1582	9615
Ilyichevsk (UA ILK)	4017	2131	3495	1798	3695	8626	4594	160	8742	1145	3421	1707	7948	4307	1614	1128	3934	3743	7583	0	344	384	4023	1836	7744	8814	1477	8526	6954	3285	8507	2075	8913	8203	13	681	5930	1115	8586	1441	3490	3108	8360	7593	7598	7603	8947	6129	7935	2153	6123	1481	1879	6769	7776	8874	9015	3424
Istanbul Ambarli (TR AMB)	3673	1792	3166	1460	3316	8274	4267	203	8367	793	3044	1373	7575	3897	1273	790	3556	3414	7201	344	0	55	3632	1490	7403	8471	1137	8223	6625	2935	8151	1731	8562	7843	347	349	5609	779	8199	1097	3135	2750	7915	7211	7216	7228	8609	5791	7591	1809	5770	1152	1542	6425	7429	8511	8670	3074
Izmit (TR IZT)	3720	1643	3198	1504	3398	8335	4303	251	8451	850	3124	1422	7657	4010	1323	837	3637	3446	7292	384	55	0	3732	1545	7453	8523	1186	8313	6663	2988	8213	1784	8622	7909	397	390	5639	824	8295	1150	3193	2814	8069	7302	7307	7281	8656	5838	7644	1856	5826	1190	1588	6478	7485	8579	8724	3127
Jebel Ali Terminal (AE JEA)	6709	4767	6187	4439	6291	5909	1887	3836	6002	2922	6019	4368	5210	6872	4268	3785	6531	6435	4836	4023	3632	3732	0	2152	5037	6105	4141	5857	4280	5907	5785	4772	6197	5477	3980	3444	3243	2857	5834	4102	6107	894	5549	4846	4851	4862	6243	3425	5261	4845	3404	4202	4519	4095	5064	6146	6304	6046
Jeddah (SA JED)	4522	2625	4000	2297	4149	6795	2788	1694	6888	746	3877	2226	6096	4730	2126	1643	4389	4248	5722	1836	1490	1545	2152	0	5924	6992	1999	6744	5152	3765	6672	2585	7083	6363	1838	1302	4130	715	6720	1960	3965	1270	6436	5732	5737	5749	7130	4312	6133	2658	4291	2015	2377	4967	5950	7032	7191	3904
Kaohsiung (TW KHH)	10430	8538	9908	8210	10062	898	3164	7607	1006	6697	9790	8139	235	10643	8039	7556	10302	10156	354	7744	7403	7453	5037	5924	0	1107	7912	849	1653	9678	790	8493	1205	481	7751	7215	1805	6628	838	7873	9878	4779	554	364	369	337	1254	1611	224	8566	1632	7923	8290	1121	166	1150	1319	9817
Kobe (JP UKB)	11500	9606	10978	9278	11130	341	4232	8675	810	7741	10858	9207	999	11711	9107	8624	11370	11226	1369	8814	8471	8523	6105	6992	1107	0	8980	394	2780	10746	801	9563	219	761	8819	8283	2873	7696	775	8941	10946	5847	729	1379	1384	1353	268	2679	917	9636	2700	8993	9358	2213	1103	961	333	10885
Koper (SI KOP)	3532	1642	3010	1290	3166	8783	4776	1340	8876	1307	2894	1202	8084	3747	1103	620	3406	3258	7710	1477	1137	1186	4141	1999	7912	8980	0	8732	7129	2792	8660	1594	9071	8352	1484	829	6118	1287	8708	103	2992	3259	8424	7720	7725	7737	9118	6300	8110	1668	6279	8	1377	6944	7938	9020	9179	2931
Kwangyang (KR KAN)	11212	9358	10733	9030	10882	72	3985	8427	437	7426	10610	8959	699	11463	8859	8376	11122	10989	1084	8526	8223	8313	5857	6744	849	394	8732	0	2525	10498	427	9275	565	436	8571	8035	2625	7448	401	8693	10698	5599	382	1094	1099	1068	614	2432	616	9348	2452	8774	9110	1905	817	588	679	10637
Laem Chabang (TH LCH)	9640	7754	9118	7429	9318	2569	2398	6821	2679	5806	9044	7340	1894	9930	7258	6772	9557	9366	1489	6954	6625	6663	4280	5152	1653	2780	7129	2525	0	8908	2407	7703	2885	2103	6967	6432	1041	5841	2527	7093	9113	3982	2251	1499	1504	1482	2920	831	1879	7776	836	7133	7508	681	1714	2773	2979	9047
Le Havre (FR LEH)	774	1155	252	1663	431	10549	6542	3136	10642	3076	163	1830	9850	1015	1997	2152	674	500	9476	3285	2935	2988	5907	3765	9678	10746	2792	10498	8908	0	10415	1219	10826	10107	3269	2614	7873	3042	10463	2732	232	5014	10179	9486	9491	9492	10873	8055	9889	1137	8034	2804	1512	8723	9693	10775	10934	171
Liangyungang (CN LYG)	11025	9286	10634	8958	10810	493	3913	8355	324	7413	10538	8887	626	11391	8787	8304	11050	10889	1012	8507	8151	8213	5785	6672	790	801	8660	427	2407	10415	0	9241	971	337	8499	7963	2553	7376	89	8621	10626	5527	258	1022	1027	996	1020	2360	641	9315	2380	8674	9038	1876	745	468	1086	10565
Malaga (ES AGP)	1951	65	1429	510	1629	9375	5343	1942	9491	1877	1355	641	8697	2241	798	973	1868	1677	8332	2075	1731	1784	4772	2585	8493	9563	1594	9275	7703	1219	9241	0	9662	8937	2088	1432	6679	1864	9335	1558	1424	3842	9109	8342	8347	8352	9696	6878	8684	87	6872	1598	338	7518	8525	9607	9764	1358
Nagoya (JP NGO)	11599	9697	11077	9369	11221	511	4324	8766	980	7828	10949	9298	1097	11802	9198	8715	11461	11325	1467	8913	8562	8622	6197	7083	1205	219	9071	565	2885	10826	971	9662	0	905	8910	8374	2965	7787	945	9032	11037	5938	893	1477	1482	1451	119	2771	1028	9735	2792	9092	9449	2322	1201	1132	188	10976
Ningbo (CN NGB)	10721	8978	10330	8650	10502	495	3604	8047	556	7109	10230	8579	318	11083	8479	7996	10742	10585	704	8203	7843	7909	5477	6363	481	761	8352	436	2103	10107	337	8937	905	0	8191	7655	2245	7068	386	8313	10318	5219	101	714	719	687	954	2051	355	9011	2072	8370	8730	1572	436	699	1020	10257
Odessa (UA ODS)	4030	2139	3508	1808	3664	8622	4615	170	8715	1157	3392	1720	7923	4245	1621	1138	3904	3756	7549	13	347	397	3980	1838	7751	8819	1484	8571	6967	3269	8499	2088	8910	8191	0	697	5957	1126	8547	1445	3480	3098	8263	7559	7564	7576	8957	6139	7948	2166	6118	1494	1889	6782	7777	8859	9018	3419
Piraeus (GR PIR)	3369	1484	2847	1153	3009	8086	4079	553	8179	607	2737	1066	7387	3590	966	483	3249	3095	7013	681	349	390	3444	1302	7215	8283	829	8035	6432	2614	7963	1432	8374	7655	697	0	5421	590	8011	790	2834	2562	7727	7023	7028	7040	8421	5603	7413	1505	5582	839	1235	6247	7241	8323	8482	2773
Port Klang (MY PKL)	8616	6744	8094	6416	8268	2677	1370	5813	2770	4854	7996	6345	1978	8849	6245	5762	8508	8342	1604	5930	5609	5639	3243	4130	1805	2873	6118	2625	1041	7873	2553	6679	2965	2245	5957	5421	0	4834	2602	6079	8084	2985	2317	1614	1619	1630	3012	193	2022	6752	173	6109	6496	856	1832	2914	3072	8023
Port Said (EG PSD)	3801	1913	3279	1585	3437	7499	3492	982	7592	31	3165	1513	6800	4018	1413	931	3677	3527	6426	1115	779	824	2857	715	6628	7696	1287	7448	5841	3042	7376	1864	7787	7068	1126	590	4834	0	7424	1248	3256	1975	7140	6436	6441	6453	7834	5016	6822	1941	4995	1294	1665	5656	6654	7736	7895	3195
Qingdao (CN TAO)	11272	9334	10750	9006	10858	467	3961	8403	251	7458	10586	8935	675	11439	8835	8352	11098	10998	1061	8586	8199	8295	5834	6720	838	775	8708	401	2527	10463	89	9335	945	386	8547	8011	2602	7424	0	8669	10674	5575	307	1071	1076	1044	994	2408	697	9408	2429	8765	9086	1963	793	394	1060	10613
Rijeka (HR RJK)	3496	1603	2974	1251	3127	8744	4737	1301	8837	1265	2855	1164	8045	3708	1064	581	3367	3222	7671	1441	1097	1150	4102	1960	7873	8941	103	8693	7093	2732	8621	1558	9032	8313	1445	790	6079	1248	8669	0	2953	3220	8385	7681	7686	7698	9079	6261	8074	1632	6240	110	1338	6908	7899	8981	9140	2892
Rotterdam (NL RTM)	584	1355	149	1863	215	10749	6742	3336	10842	3277	123	2030	10050	809	2197	2372	467	305	9676	3490	3135	3193	6107	3965	9878	10946	2992	10698	9113	232	10626	1424	11037	10318	3480	2834	8084	3256	10674	2953	0	5214	10379	9686	9691	9692	11073	8255	10094	1342	8234	3009	1712	8928	9893	10975	11134	64
Salalah (OM SLL)	5626	3885	5234	3557	5409	5650	1633	2954	5743	2013	5137	3486	4951	5990	3386	2903	5649	5490	4577	3108	2750	2814	894	1270	4779	5847	3259	5599	3982	5014	5527	3842	5938	5219	3098	2562	2985	1975	5575	3220	5214	0	5291	4587	4592	4604	5985	3167	4970	3915	3146	3274	3637	3796	4805	5887	6046	5164
Shanghai (CN SHA)	11046	9050	10524	8722	10574	445	3677	8119	477	7181	10302	8651	390	11155	8551	8068	10814	10772	776	8360	7915	8069	5549	6436	554	729	8424	382	2251	10179	258	9109	893	101	8263	7727	2317	7140	307	8385	10379	5291	0	786	791	760	942	2124	421	9182	2144	8539	8802	1687	509	621	1008	10329
Shenzhen Chiwan (CN CWN)	10279	8393	9757	8068	9957	1150	3037	7460	1266	6580	9683	7990	472	10569	7897	7364	10196	10005	10	7593	7211	7302	4846	5732	364	1379	7720	1094	1499	9486	1022	8342	1477	714	7559	7023	1614	6436	1071	7681	9686	4587	786	0	5	56	1526	1420	481	8415	1441	7772	8098	937	297	1382	1592	9625
Shenzhen Da Chan Bay (CN SAD)	10284	8398	9762	8073	9962	1155	3042	7465	1271	6585	9688	7995	477	10574	7902	7369	10201	10010	15	7598	7216	7307	4851	5737	369	1384	7725	1099	1504	9491	1027	8347	1482	719	7564	7028	1619	6441	1076	7686	9691	4592	791	5	0	61	1531	1425	486	8420	1446	7777	8103	942	302	1387	1597	9630
Shenzhen Yantian (CN YTN)	10289	8363	9701	8035	9887	1120	2990	7432	1212	6590	9615	7964	420	10468	7864	7381	10127	9956	46	7603	7228	7281	4862	5749	337	1353	7737	1068	1482	9492	996	8352	1451	687	7576	7040	1630	6453	1044	7698	9692	4604	760	56	61	0	1500	1436	446	8425	1457	7741	8115	947	270	1356	1565	9642
Shimizu (JP SMZ)	11633	9744	11111	9416	11268	560	4371	8813	1030	7870	10996	9345	1146	11849	9245	8762	11508	11359	1516	8947	8609	8656	6243	7130	1254	268	9118	614	2920	10873	1020	9696	119	954	8957	8421	3012	7834	994	9079	11073	5985	942	1526	1531	1500	0	2818	1064	9769	2839	9126	9496	2355	1250	1181	96	11023
Singapore (SG SIN)	8815	6926	8293	6598	8450	2483	1553	5995	2576	5036	8178	6527	1784	9031	6427	5944	8690	8541	1410	6129	5791	5838	3425	4312	1611	2679	6300	2432	831	8055	2360	6878	2771	2051	6139	5603	193	5016	2408	6261	8255	3167	2124	1420	1425	1436	2818	0	1812	6951	20	6308	6678	646	1638	2720	2879	8205
Taipei (TW TAP)	10454	8735	10099	8410	10299	716	3379	7802	851	6842	10025	8332	150	10911	8239	7753	10538	10347	471	7935	7591	7644	5261	6133	224	917	8110	616	1879	9889	641	8684	1028	355	7948	7413	2022	6822	697	8074	10094	4970	421	481	486	446	1064	1812	0	8757	1846	8114	8489	1312	222	996	1134	10028
Tangiers (MA TNG)	1869	32	1347	585	1547	9448	5416	2020	9564	1951	1273	714	8770	2159	871	1047	1786	1595	8405	2153	1809	1856	4845	2658	8566	9636	1668	9348	7776	1137	9315	87	9735	9011	2166	1505	6752	1941	9408	1632	1342	3915	9182	8415	8420	8425	9769	6951	8757	0	6945	1672	411	7591	8598	9681	9837	1276
Tanjung Pelepas (MY TPP)	8809	6905	8246	6577	8429	2504	1532	5974	2597	5025	8157	6506	1805	9010	6406	5923	8669	8502	1431	6123	5770	5826	3404	4291	1632	2700	6279	2452	836	8034	2380	6872	2792	2072	6118	5582	173	4995	2429	6240	8234	3146	2144	1441	1446	1457	2839	20	1846	6945	0	6286	6657	680	1659	2741	2899	8184
Trieste (IT TRS)	3536	1650	3014	1301	3214	8805	4773	1348	8921	1310	2940	1215	8127	3826	1111	625	3453	3262	7762	1481	1152	1190	4202	2015	7923	8993	8	8774	7133	2804	8674	1598	9092	8370	1494	839	6109	1294	8765	110	3009	3274	8539	7772	7777	7741	9126	6308	8114	1672	6286	0	1388	6948	7955	9039	9194	2943
Valencia (ES VLC)	2275	382	1753	161	1907	9161	5154	1745	9254	1683	1635	331	8462	2488	508	758	2147	2001	8088	1879	1542	1588	4519	2377	8290	9358	1377	9110	7508	1512	9038	338	9449	8730	1889	1235	6496	1665	9086	1338	1712	3637	8802	8098	8103	8115	9496	6678	8489	411	6657	1388	0	7323	8316	9398	9557	1672
Vung Tau (VN SGN)	9455	7569	8933	7244	9133	2005	2213	6636	2119	5668	8859	7166	1327	9745	7073	6587	9372	9181	927	6769	6425	6478	4095	4967	1121	2213	6944	1905	681	8723	1876	7518	2322	1572	6782	6247	856	5656	1963	6908	8928	3796	1687	937	942	947	2355	646	1312	7591	680	6948	7323	0	1150	2242	2418	8862
Xiamen (CN XMN)	10462	8564	9940	8236	10088	869	3191	7633	961	6674	9816	8165	168	10669	8065	7582	10328	10188	287	7776	7429	7485	5064	5950	166	1103	7938	817	1714	9693	745	8525	1201	436	7777	7241	1832	6654	793	7899	9893	4805	509	297	302	270	1250	1638	222	8598	1659	7955	8316	1150	0	1105	1316	9843
Xingang (CN TXG)	11391	9646	10999	9318	11170	654	4273	8715	188	7779	10898	9247	987	11751	9147	8664	11410	11255	1372	8874	8511	8579	6146	7032	1150	961	9020	588	2773	10775	468	9607	1132	699	8859	8323	2914	7736	394	8981	10975	5887	621	1382	1387	1356	1181	2720	996	9681	2741	9039	9398	2242	1105	0	1246	10925
Yokohama (JP YOK)	11701	9805	11179	9477	11329	626	4432	8874	1095	7941	11057	9406	1211	11910	9306	8823	11569	11427	1582	9015	8670	8724	6304	7191	1319	333	9179	679	2979	10934	1086	9764	188	1020	9018	8482	3072	7895	1060	9140	11134	6046	1008	1592	1597	1565	96	2879	1134	9837	2899	9194	9557	2418	1316	1246	0	11084
Zeebrugge (BE ZEE)	627	1294	87	1802	269	10688	6681	3275	10781	3212	83	1969	9989	861	2136	2311	519	347	9615	3424	3074	3127	6046	3904	9817	10885	2931	10637	9047	171	10565	1358	10976	10257	3419	2773	8023	3195	10613	2892	64	5164	10329	9625	9630	9642	11023	8205	10028	1276	8184	2943	1672	8862	9843	10925	11084	0

Table B.7: Natural order of ports in the MAE data set

Number	Port
1	Yokohama
2	Shimizu
3	Nagoya
4	Kobe
5	Busan
6	Kwangyang
7	Dalian
8	Xingang
9	Qingdao
10	Liangyungang
11	Shanghai
12	Ningbo
13	Fuzhou
14	Taipei
15	Xiamen
16	Kaohsiung
17	Shenzhen Yantian
18	Hong Kong
19	Shenzhen Chiwan
20	Shenzhen Da Chan Bay
21	Vung Tau
22	Laem Chabang
23	Singapore
24	Tanjung Pelepas
25	Port Klang
26	Colombo
27	Jebel Ali
28	Salalah
29	Jeddah
30	Port Said
31	Damietta
32	Izmit
33	Istanbul Ambarli
34	Odessa
35	Ilyichevsk
36	Constantza
37	Piraeus
38	Rijeka
39	Koper
40	Trieste
41	Gioia Tauro
42	Genoa
43	Fos
44	Barcelona
45	Valencia
46	Malaga
47	Algeciras
48	Tangiers
49	Le Havre
50	Felixstowe
51	Zeebrugge
52	Antwerp
53	Rotterdam
54	Bremerhaven
55	Hamburg
56	Gothenburg
57	Aarhus
58	Gdansk

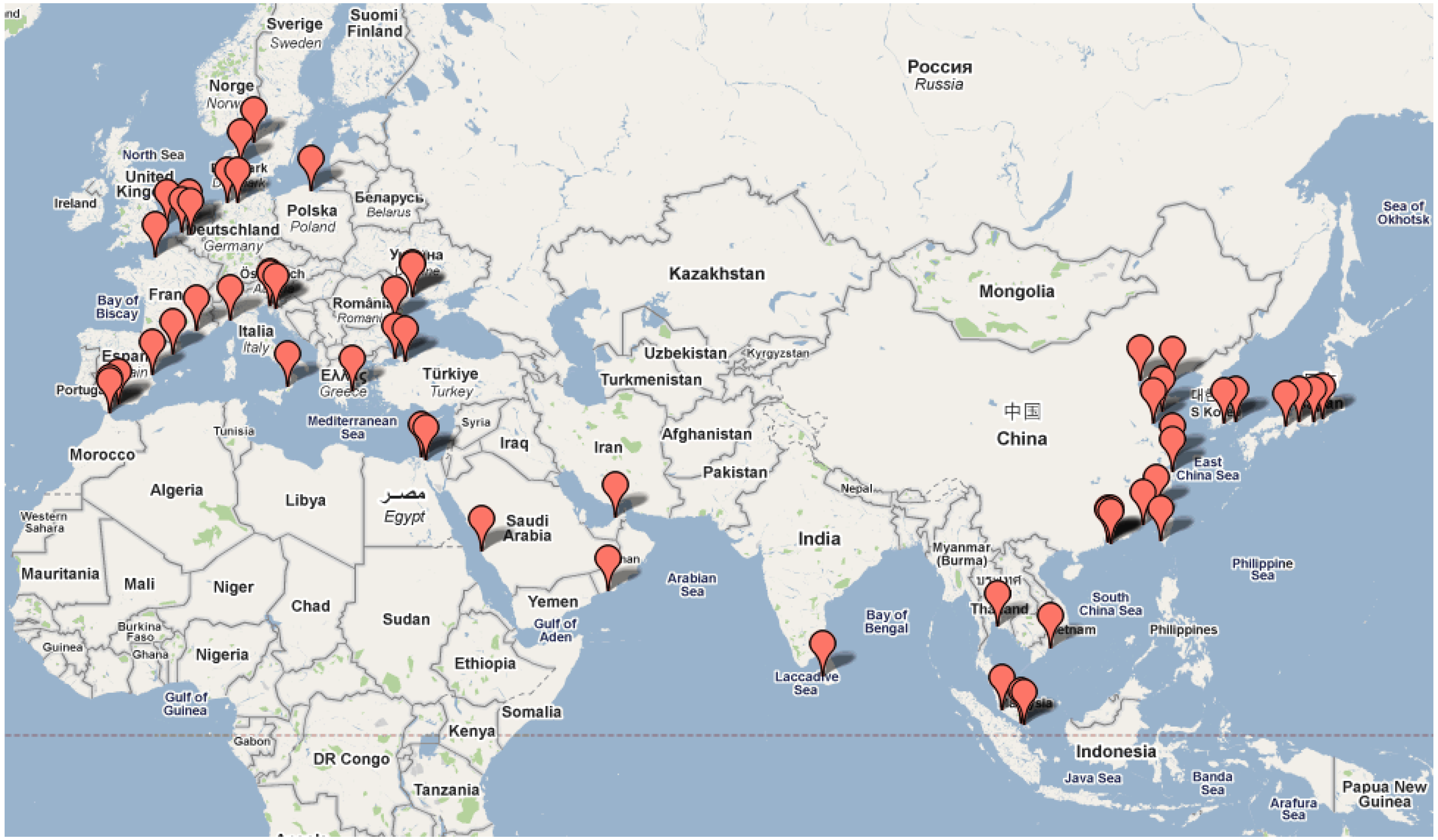


Figure B.1: Map of ports in MAE data set (© Google Maps)

Appendix C

Best network

In this appendix chapter we present additional information regarding the best service network we found during the benchmark study. First, page 196 specifies the full service network consisting of 8 routes. Then, on the next page, Table C.2 provides a comparison between the original Maersk service network, and our best network. The comparison focuses on the number of port occurrences in both networks.

Overview of routes in the optimal network

Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8
Yokohama	Nagoya	Kobe	Shimizu	Yokohama	Yokohama	Yokohama	Kwangyang
Busan	Kaohsiung	Kwangyang	Xiamen	Kobe	Qingdao	Kwangyang	Qingdao
Port Said	Genoa	Dalian	Kaohsiung	Busan	Shanghai	Shanghai	Ningbo
Malaga	Le Havre	Xingang	Shenzhen Yantian	Kwangyang	Xiamen	Vung Tau	Taipei
Rotterdam	Felixstowe	Qingdao	Tanjung Pelepas	Liangyungang	Singapore	Laem Chabang	Shenzhen Yantian
Aarhus	Antwerp	Liangyungang	Salalah	Fuzhou	Tanjung Pelepas	Singapore	Hong Kong
Gothenburg	Rotterdam	Genoa	Port Said	Koahsiung	Port Klang	Salalah	Colombo
Bremerhaven	Gothenburg	Barcelona	Piraeus	Shenzhen Chiwan	Salalah	Port Said	Jebel Ali
Felixstowe	Bremerhaven	Le Havre	Koper	Shenzhen Da Chan Bay	Damietta	Damietta	Jeddah
Valencia	Le Havre	Bremerhaven	Barcelona	Vung Tau	Piraeus	Istanbul Ambarli	Piraeus
Barcelona	Genoa	Gothenburg	Valencia	Port Klang	Koper	Piraeus	Gioia Tauro
Fos	Odessa	Gdansk	Zeebrugge	Port Said	Valencia	Trieste	Fos
Trieste	Singapore	Aarhus	Antwerp	Istanbul Ambarli	Algeciras	Gioia Tauro	Antwerp
Koper	Vung Tau	Hamburg	Tangiers	Gioia Tauro	Tangiers	Bremerhaven	Hamburg
Piraeus	Hong Kong	Malaga	Algeciras	Genoa	Zeebrugge	Hamburg	Aarhus
Istanbul Ambarli	Xiamen	Valencia	Genoa	Malaga	Hamburg	Rotterdam	Gdansk
Jeddah	Fuzhou	Fos	Gioia Tauro	Antwerp	Rotterdam	Antwerp	Damietta
Salalah	Ningbo	Gioia Tauro	Rijeka	Hamburg	Zeebrugge	Zeebrugge	Vung Tau
Colombo	Liangyungang	Constantza	Salalah	Rotterdam	Rijeka	Felixstowe	Shenzhen Yantian
Port Klang	Qingdao	Istanbul Ambarli	Tanjung Pelepas	Zeebrugge	Izmit	Trieste	Kaohsiung
Tanjung Pelepas	Xingang	Port Said	Xiamen	Le Havre	Colombo	Constantza	Xiamen
Laem Chabang	Dalian	Salalah	Ningbo	Tangiers	Tanjung Pelepas	Izmit	Shanghai
Shenzhen Da Chan Bay	Kwangyang	Jebel Ali	Qingdao	Algeciras	Shenzhen Da Chan Bay	Jeddah	Qingdao
Shenzhen Chiwan	Busan	Port Klang	Kobe	Piraeus	Shenzhen Chiwan	Jebel Ali	Xingang
Hong Kong		Tanjung Pelepas		Salalah	Xiamen	Liangyungang	
Shenzhen Yantian		Singapore		Jebel Ali	Fuzhou	Nagoya	
Taipei		Vung Tau		Colombo	Qingdao	Shimizu	
Liangyungang		Ningbo		Port Klang	Xingang		
Xingang		Shanghai		Hong Kong	Dalian		
Kwangyang		Busan		Shenzhen Yantian			
Busan Nagoya				Fuzhou			

Table C.2: Comparison of port occurrence in Maersk network & our best network

Port name	Occ. Maersk[(1)]	Occ. Best[(2)]	**Occurs in routes**
Aarhus	1	3	[1,3,8]
Algeciras	2	3	[4,5,6]
Antwerp	1	5	[2,4,5,7,8]
Barcelona	1	3	[1,3,4]
Bremerhaven	5	4	[1,2,3,7]
Busan	3	5	[1,1,2,3,5]
Colombo	2	4	[1,5,6,8]
Constantza	1	2	[3,7]
Dalian	2	3	[2,3,6]
Damietta	4	3	[6,7,8]
Felixstowe	4	3	[1,2,7]
Fos	1	3	[1,3,8]
Fuzhou	1	4	[2,5,5,6]
Gdansk	1	2	[3,8]
Genoa	1	5	[2,2,3,4,5]
Gioia Tauro	2	5	[3,4,5,7,8]
Gothenburg	1	3	[1,2,3]
Hamburg	3	5	[3,5,6,7,8]
Hong Kong	9	4	[1,2,5,8]
Ilyichevsk	1	0	-
Istanbul Ambarli	1	4	[1,3,5,7]
Izmit	1	2	[6,7]
Jebel Ali	1	4	[3,5,7,8]
Jeddah	3	3	[1,7,8]
Kaohsiung	1	4	[2,4,5,8]
Kobe	1	3	[3,4,5]
Koper	1	3	[1,4,6]
Kwangyang	1	6	[1,2,3,5,7,8]
Laem Chabang	1	2	[1,7]
Le Havre	2	4	[2,2,3,5]
Liangyungang	1	5	[1,2,3,5,7]
Malaga	1	3	[1,3,5]
Nagoya	2	3	[1,2,7]
Ningbo	4	4	[2,3,4,8]
Odessa	1	1	[2]
Piraeus	1	6	[1,4,5,6,7,8]
Port Klang	8	5	[1,3,5,5,6]
Port Said	6	5	[1,3,4,5,7]
Qingdao	2	7	[2,3,4,6,6,8,8]
Rijeka	1	2	[4,6]
Rotterdam	5	5	[1,2,5,6,7]
Salalah	3	7	[1,3,4,4,5,6,7]
Shanghai	7	4	[3,6,7,8]
Shenzhen Chiwan	3	3	[1,5,6]
Shenzhen Da Chan Bay	1	3	[1,5,6]
Shenzhen Yantian	8	5	[1,4,5,8,8]
Shimizu	1	2	[4,7]
Singapore	4	4	[2,3,6,7]
Taipei	1	2	[1,8]
Tangiers	4	3	[4,5,6]
Tanjung Pelepas	10	6	[1,3,4,4,6,6]
Trieste	1	3	[1,7,7]
Valencia	1	4	[1,3,4,6]
Vung Tau	1	5	[2,3,5,7,8]
Xiamen	2	6	[2,4,4,6,6,8]
Xingang	2	5	[1,2,3,6,8]
Yokohama	2	4	[1,5,6,7]
Zeebrugge	2	5	[4,5,6,6,7]
Total occurrences	144	221	

(1) Corresponds to the number of visits per port in the original service network.
(2) Corresponds to the number of visits per port in our best service network.