Solving the Twin Hub Bundling problem

Master's Thesis

Sebastiaan Meijer

Solving the Twin Hub Bundling problem

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Sebastiaan Meijer
born in Heemstede, the Netherlands

TU Delft

Algorithmics Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
alg.ewi.tudelft.nl

OTB Research Institute
Jaffalaan 9
Delft, the Netherlands
otb.tudelft.nl

Solving the Twin Hub Bundling problem

Author: Sebastiaan Meijer
Student id: 1374168
Email: sjjmeijer@hotmail.com

Abstract

Optimisation of transport networks can yield substantial reductions in transport costs and improve the quality of service for the users of these networks. When optimising intermodal rail-road networks, organising large transport volumes on transport links is key for bringing down the costs per container-kilometre. By bundling container flows in Twin Hub Bundling (THB) networks, thereby integrating the flows of the Dutch and Belgian seaports, the Twin Hub project aims to stimulate intermodal rail-road transport in North-West Europe. The goal of this thesis is to create a bundling tool that finds the most promising THB networks based on projected container flows. In this thesis, a modelling is presented of the THB problem, introducing three innovative features: the organisation of intermodal transport services in batches, the introduction of indirect trains, and the integration of train and truck services in one service network. We then prove this problem to be an NP-hard optimisation problem, and present an exact algorithm based on Integer Linear Programming (ILP) as well as a relaxation of that formulation. An experimental evaluation shows that both the exact algorithm and the combination of the relaxation and the ILP algorithm (cascade algorithm) are capable of producing good-quality solutions to real-world sized instances within 24 hours. Finally, we present the bundling tool itself, which provides an easy interface for running the algorithms, evaluating the results, and selecting the most promising THB network to be implemented in a pilot operation.

Thesis Committee:

Chair:	Prof. Dr. C. Witteveen, Faculty EEMCS, TU Delft
University supervisor:	Prof. Dr. C. Witteveen, Faculty EEMCS, TU Delft
Company supervisor:	Dr. E.D. Kreutzberger, OTB Research Institute
Committee Members:	Dr. M.M. de Weerdt, Faculty EEMCS, TU Delft
	Ir. H.J.A.M. Geers, Faculty EEMCS, TU Delft

Preface

For my Master's thesis at the Algorithmics group at the Delft University of Technology, I have worked on a practical assignment for the OTB Research Institute as part of the Twin Hub project. The practical setting of my thesis meant that there was a clear goal to work towards: the development of a tool which bundles flows on trains in so-called Twin Hub Bundling (THB) Networks.

Naturally, developing this bundling tool involved more than just implementing an existing algorithm. Even though the Twin Hub concept is very innovative, I did not expect that there would be so little existing research solving other intermodal transport problems. This thesis therefore includes all steps from the analysis of the problem to the implementation of the algorithm and the tool, making it a much larger project than I initially expected.

I would like to thank Prof. dr. C. Witteveen for introducing me to this project, and for giving suggestions that pointed me in the right direction in my research and while writing this thesis. Even though some suggestions required a significant rewrite of this thesis, it has certainly improved the structure and overall quality of this document.

I also want to thank Dr. E.D. Kreutzberger for sharing his expertise on transportation research in general, and on the Twin Hub project in particular. After every meeting, I had the feeling I had a better understanding of the goals and structure of the Twin Hub project, and how I could continue with the modelling or the implementation of the tool.

Then I would also like to thank my fellow students in HB 07.130, who not only helped me by giving some useful suggestions for my thesis, but who also provided some much-needed distraction at times.

A special thanks go out to my parents and my brother for their support during my thesis.

Finally, I would like to thank everyone else who have made suggestions for improving my research, offered helpful tips on how to structure my thesis, or those who have supported me at times when things did not go as planned. I hope you'll enjoy reading the result.

Sebastiaan Meijer
Delft, the Netherlands
21st November 2012

Contents

List of Figures

Chapter 1

Introduction

For centuries The Netherlands has been a central trade hub with goods being imported from all over the globe. With its strategic location on the North Sea at the deltas of two of Europe's major rivers, The Netherlands still plays a key role in the transportation of goods to and from North-West Europe. For example, the import from and export to Germany, Belgium, the UK, and France together was worth 193.7 billion euros and 127.3 billion euros in 2010 respectively [36], and accounts for half the total imports and exports for The Netherlands. Re-export of imported goods accounts for 41% of all Dutch exports [12], and makes The Netherlands the most important trade partner of Germany (9.9% of its total trade volume), the second most important trade partner of Belgium (15.9% of its total trade volume), the third most important trade partner of the UK (7.0% of its total trade volume), and the fifth most important trade partner of France (6.0% of its total trade volume) [35].

Trade in North-West Europe not only benefits the transport companies that transport the goods throughout the region, but cheap and efficient transportation is also vital to the economy, as it allows companies to cut the costs of their products, giving them a competitive edge in an increasingly globalising market. The cost of transportation of a product from its place of manufacturing to its destination amounts to 10-15% of the product's price [6] for European manufacturers. With total trade volumes in North-West Europe worth 321 billion euros, and the transport costs thus amounting to more than 32 billion, saving just 1% in transport costs would amount to a substantial cost savings of 300 to 450 million euros.

To create a more efficient efficient transportation system, large and costly infrastructure projects are executed to facilitate the growth in global trade. New roads are being built to ease congestion on the motorways, the Betuweroute – a dedicated freight railway from Rotterdam to Germany – was opened in 2007, giving German long freight trains a direct connection to the Port of Rotterdam, and the second Maasvlakte is currently being built to handle the increasing trade volumes at the Port of Rotterdam.

These multi-billion euro infrastructure projects are required to maintain and possibly expand the current market position of The Netherlands as a central trade hub for North-West Europe. When deciding what projects lead to the most efficiency gain, the Dutch government not only investigates where the bottlenecks in the transport system are, but also looks at other, external factors. Transportation by road, for example, leads

Figure 1.1: A freight train entering the tunnel towards the Kijfhoek shunting yard on the Betuweroute (source: ANP)

to relatively high CO_2-emissions per tonne-kilometre and reduces the air quality near the motorways. Trains can offer a more environmentally friendly way of transporting cargo, while at the same time reducing the congestion on the road.

Making these alternative options more competitive is not a simple matter of creating the required infrastructure: companies may decide to continue using trucks, even though using trains may be cheaper. Transporting cargo by truck offers, for example, greater flexibility in terms of volume to be transported and transport frequency, as a train requires a high utilisation rate (i.e. a large number of wagons attached to the train) in order for it to be cheaper than road transport. Additionally, trains are bound to fixed schedules, offering less flexibility in departure and arrival times.

In order to to make rail freight transportation more competitive, high-frequency train services need to be organised to increase the flexibility for the companies that may use these services. With high-frequency services and lower transportation costs, rail transport will become more competitive. Keeping the utilisation rate of these trains high, however, is a major challenge, as high-frequency services require a large and steady supply of cargo to make them profitable. Even for very large terminals, such as the Port of Rotterdam – the largest seaport by volume in Europe, with 40% of the total Dutch trade volume being handled by this port alone [32, 34] – the flows between Rotterdam and many terminals throughout North-West Europe are too small to offer a daily transport service [24].

The EU has created several incentives for transportation companies to develop profitable freight train services throughout Europe, with the aim of better utilisation of the available infrastructure, and reducing the amount of transport-related CO_2-emissions. The Marco Polo Programme [13], for example, stimulated the creation of intermodal transport services by offering financial support in the start-up phase of these services. Intermodal freight services combine multiple modes of transport (e.g. road and rail transport) to deliver cargo. This cargo consists of shipping containers that are collected using trucks, and then placed on a train. The train then rides to a rail terminal near the destination of the containers, and these containers are then distributed

to their destinations using trucks.

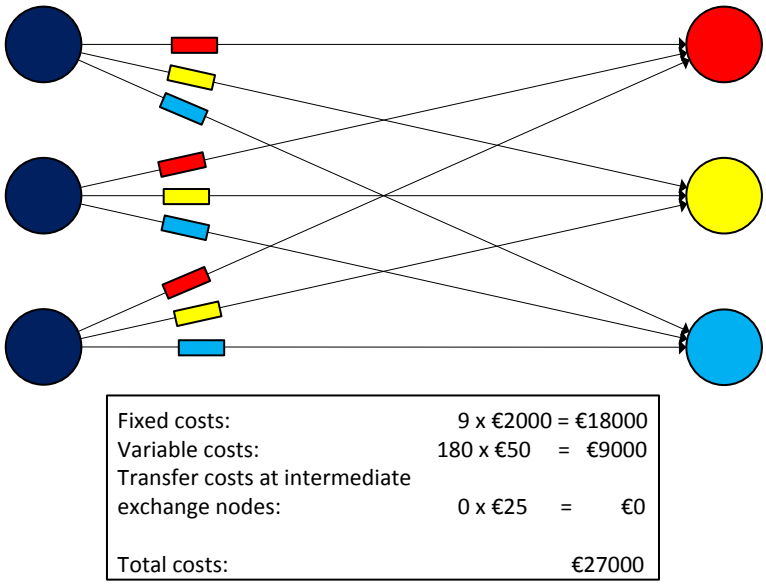

Figure 1.2: Transportation of containers on direct trains.

Despite these incentives, intermodal rail-road freight transport has not taken off as expected, largely due to the relatively small point-to-point flows between terminals, which do not allow for creating profitable high-frequency train services [24]. Figure 1.2 illustrates this problem: a high-frequency direct train service (i.e. transporting containers from origin to destination) requires a large number of trains to transport all containers from the source terminals to the destination terminals. Each train in this example transports only a relatively small number of containers: 20 containers on a train with a capacity of 60, resulting in a low loading degree. This means that the fixed costs per train (e.g. the costs for the locomotive and personnel) are amortised over few transported containers, resulting in a high cost per load unit, which decreases the competitiveness of the service. Waiting until enough containers have accumulated at the source terminals is not a viable option either, as this lowers the service frequency, which impacts the competitiveness of the train services too.

To solve this problem, a service network needs to be designed with the following two properties:

1. The trains need have a high loading degree

2. The transport services must provide a high service frequency between source and destination terminals.

Creating a network with these two properties is the challenge the Twin-Hub Network project aims to solve.

1.1 Twin Hub project

The Twin Hub Network project is a collaborative project between seaports, intermodal railway operators, and universities and aims to stimulate intermodal rail freight transport in North-West Europe by creating a service network that is profitable for all parties involved. The trains in this new service network will transport containers arriving at the Dutch and Belgian seaports to destinations throughout North-West Europe (and vice versa). In the current situation, road transport has a large market share, with intermodal transport quality being poor in terms of network connectivity and service frequency, except for transport in some large flow corridors, from and to some large nodes, and in some well-organised regions [3].

The goal of the project is to create the conditions to shift the container flows in North-West Europe from road to rail transport. This means that competitive door-to-door transport services between the Dutch and Belgian seaports and inland destinations must be offered, with an acceptable level of service, acceptable transfer speed, and desirable departure and arrival times of trains. When these conditions are met, a shift in flows is expected, resulting in a larger market share for intermodal rail transport. This shift makes transport more sustainable, regional accessibility more robust, regions more competitive and increases territorial and economic cohesion within Europe.

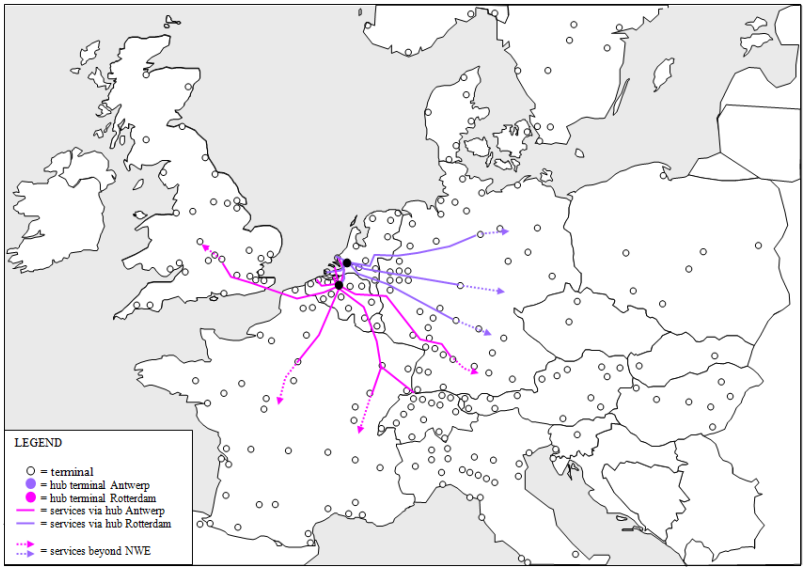

Figure 1.3: The potential service area for the Twin Hub Network project.

Achieving this goal requires a well-designed service network offering high-quality door-to-door transport services, while reducing the operational costs compared to the current situation. The potential service area for this network is large (Figure 1.3), which means that the potential savings are also high. There are a number of measures that can be taken to make this possible: the bundling of flows, and the creation of 'Twin Hub Networks'. Both concepts will be explained below.

1.1.1 Bundling networks

Bundling is the process of organising sufficiently large train loads for flows on the required level of service. Bundling flows can be achieved in three different ways: The first, bundling in time by reducing the transport frequency, is unwanted, as this reduces the quality of the service offered. The second, bundling of categories, for example by combining intermodal flows with non-intermodal flows such as bulk goods, is only a viable option on some connections where such bundling can be organised easily. The third option is the bundling of directions. This means that flows belonging to different transport relations move in the same trains during a part of their journey.

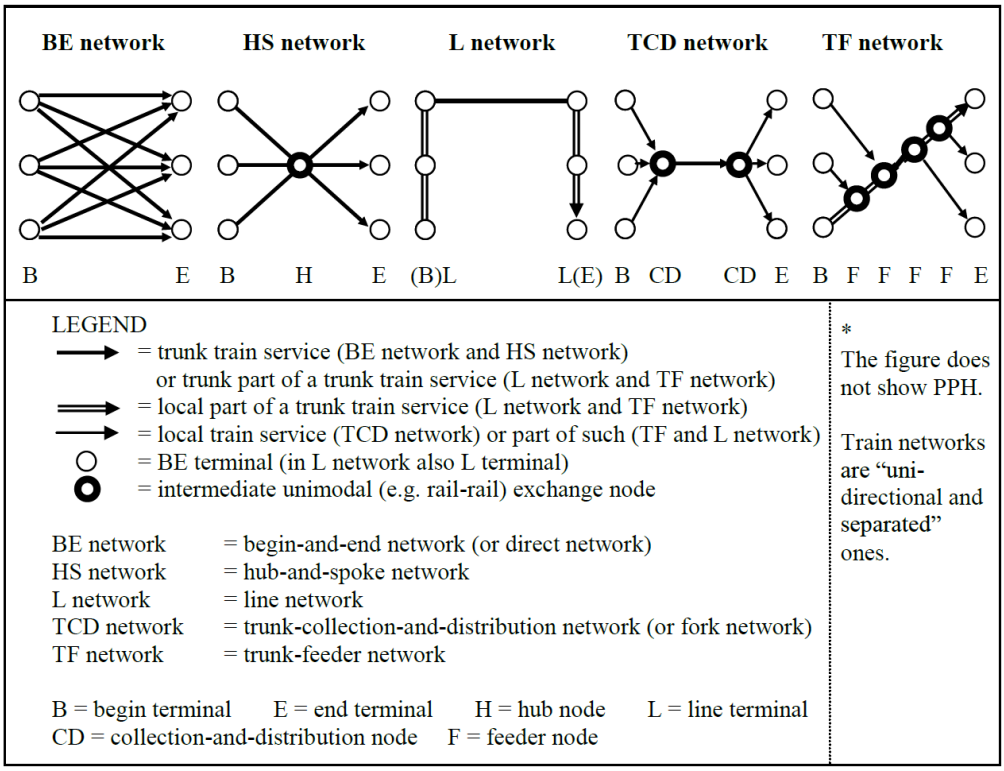

Figure 1.4: Different types of bundling [27].

The bundling of directions is the most important way of bundling, as it is the most promising way of creating sufficiently large flows to achieve a high loading degree on the trains, yielding a profitable service. Figure 1.4 shows the different ways in which direction bundling can be applied.

We distinguish two types of bundling: direct bundling, like in the left network shown in Figure 1.4, and complex bundling, like in the other networks shown in Figure 1.4. Direct bundling is the best solution for large flows which allow for filling a train at the desired service frequency. These networks do not have intermediate exchange nodes and therefore have the shortest train routes and a lower cost compared to complex bundling networks. For smaller flows, direct bundling may not be reasonable, as this results in small train loads. This in turn results in high transportation costs per load unit, making the service unprofitable or uncompetitive. Applying complex bund-

ling to aggregate the flows from different begin terminals to different end terminals on a single train, a larger train load may be organised, allowing the fixed costs per train to be amortised over more containers, driving down the transport cost per load unit.

We distinguish four types of complex bundling networks: hub-and-spoke (HS) networks, line (L) networks, trunk-collection-and-distribution (TCD) networks, and trunk-feeder (TF) networks. Each of these network types have different characteristics concerning train utilisation, detour length, and the additional exchange at intermediate nodes [26]. The analysis by Kreutzberger et al. [27] shows that HS networks work best for medium network transport volumes, as these have relatively small detours, no local networks, and a limited amount of exchange at intermediate nodes. For small network transport values, L networks work best, as the total number of transshipments is the same as for the direct network.

In the Twin Hub project, a hub-and-spoke network is expected to offer the greatest benefits, as the consolidation of small flows at the hub allows for larger train loads to be transported using the same amount of trains, or a higher service frequency with the same sized train loads. Another benefit is that smaller transport network volumes are required to create a profitable service network.

1.1.2 Flow bundling techniques

The bundling operations can take place in alternative ways. The three most important operational and technical options are [1, p. 46-55]:

1. The exchange of single wagons between trains (along with their containers). The hub then needs to be a gravity shunting yard.

2. The exchange of wagon groups between trains (along with their containers). The hub can then either be a gravity shunting yard or a flat shunting yard.

3. The exchange of containers between trains. This type of exchange requires a terminal employing cranes or other transshipment equipment.

In the first type, arriving trains are pushed up the hill of the shunting yard, after which individual wagons are decoupled. The wagons then roll down the hill by gravity alone towards an array of parallel tracks connected by switches. The new trains are formed by correctly setting the switches for each wagon. The disadvantages of this method are a relatively high cost per load unit involved, and a relatively long service time for a batch of trains. Exchanging individual load units causes high dwell times at the hub: measured dwell times for exchanging individual wagons at a gravity shunting yard are on average 3-6 hours [23], dependent on the batch size, with peaks of almost 12 hours. A detailed break-down of the process, including the time each process step takes is given in [1, p. 98-99].

The sorting of wagon groups can take place at gravity shunting yards or flat shunting yards. In gravity shunting yards, the process is similar to shunting individual wagons. In flat shunting yards, wagon groups are decoupled from the original trains and sorted using locomotives to push the wagon group to the correct track. Sorting wagon groups is much faster than sorting individual wagons, with dwell times as short as 0.8 hours, with a maximum dwell time of 2.6 hours in the German turntable network [23].

The disadvantage still remains that wagon group exchange is only suited for the wagon group market: integrating load units arriving on trucks remains difficult.

The exchange of wagon groups can either be done at gravity shunting yards or flat shunting yards. In both cases, faster transfer times and lower costs are achieved than when shunting individual wagons, as fewer decoupling, moving, and coupling actions are required to form trains of a similar length. However, the disadvantage remains that this type of shunting is only suitable for the wagon group market.

Figure 1.5: Intermodal container transshipment using overhead cranes [5].

In new hub terminals, the arriving trains are not decoupled into individual wagons or wagon groups. Instead, individual containers are transferred from one wagon onto another – empty – wagon by overhead cranes. Figure 1.5 shows such a new hub terminal. While this may seem inefficient when dealing with large directional wagon groups, this method has the advantage that the train configurations are not changed; meaning that shunting costs and the costs of delays caused by (de)coupling wagons, filling air tubes, and brake tests are avoided [1, p. 127]. Dwell times in these types of networks are generally less than 1 hour, with some trains taking up to two hours [23]. Other advantages of this type of exchange include low transshipment costs and increased flexibility, as containers transported by other methods than train wagons (e.g. trucks) can be easily incorporated in the transshipment process. Rotter [33] explains the advantages of using this type of hub terminal in more detail.

When using a new hub terminal, the wagons of an incoming train do not have to be disconnected to participate in the bundling process, meaning that containers arriving at the hub do not necessarily have to take part in the sorting process. This creates the opportunity for letting trains continue their journey to a destination terminal instead of unloading all their containers like in a shuttle service. Figure 1.6 shows the service network with these trains that ride through the hub (‘indirect trains’). This means less trains are required to operate a network with comparable performance in terms of network transport volume, service frequency, and network connectivity. Where train loads in the direct network were only 20 containers per train (Figure 1.2), now 60

containers can be transported on each indirect train. When several of these indirect trains meet, some containers can be left on the train, reducing the time required at the hub for transshipment and cutting transshipment costs. If all transport relations in a network have the same transport volume f, the average number of transshipped containers on each train is $f * \frac{n-1}{n}$, with n the number of destinations. In the example shown in Figure 1.6, only ⅔ of the containers of the three network trains (i.e. 40 containers per train) need to be exchanged. This process is most efficient when these indirect trains meet simultaneously at the hub in 'batches', as containers can then be transferred between trains instead of being put on the stack to be retrieved later. This optimises the capacity of the hub.

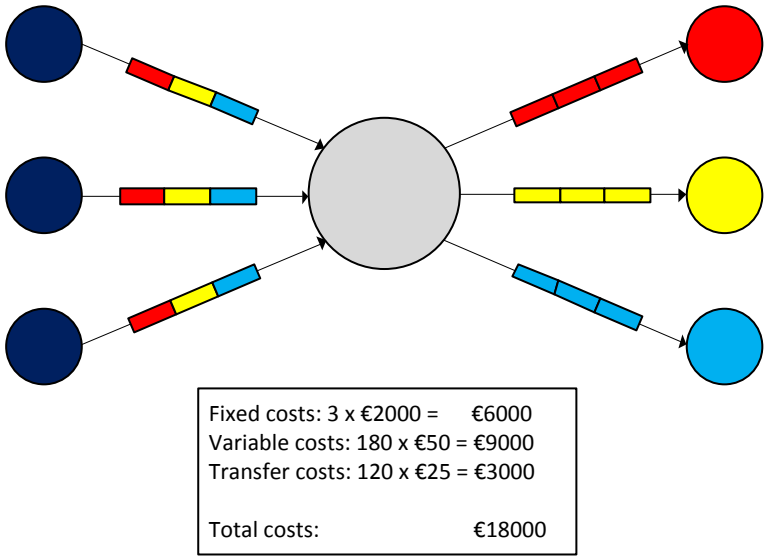

Figure 1.6: Using indirect trains in a hub-and-spoke network.

A true hub terminal is designed to facilitate larger amounts of rail-rail transshipment, and has a different layout than a rail-road terminal. It has more tracks beneath the cranes and a sorting and internal transport system to optimize the transfer of containers between different batches. This type of terminal is, however, only required when the Twin Hub project becomes a success in the long term. In the short and medium term ($<$ 10 years) any type of transshipment can be used to transfer containers [25]. It must be noted, however, that a slower transshipment method will likely cause a lower modal shift from road to rail as longer transport times make the service less competitive. A survey of several new hub terminal concepts was carried out in 1999 by Bontekoning and Kreutzberger [2].

1.1.3 The Twin Hub Network

The Twin Hub Network is an innovative concept involving the two major seaports in North-West Europe, Antwerp and Rotterdam, and smaller seaports on the North Sea

coast between Amsterdam and Zeebrugge. By bundling the flows from more than one seaport, an additional benefit is realised compared to using hub-and-spoke bundling alone. This way, the size of the train loads, the service frequency, and the network connectivity increase. The increased network connectivity means that more of the smaller seaports and European inland terminals can be accessed than would by possible by using hub-and-spoke bundling or other complex bundling types for one seaport alone.

The Twin Hub Network acknowledges the fact that the seaports are competitors, so the focus of the extension of the service area – Rotterdam now services Belgian containers, and vice versa – is focused on those areas where the ports complement each other. Belgian flows towards the South-Western corridor (e.g. France), where Antwerp has a strong market position, will still move on Belgian trains. Similarly, Dutch flows towards the North-Eastern corridor (e.g. Germany, Poland), where Rotterdam has a strong market position, will still be transported on Dutch trains.

Hub-and-spoke bundling is the central bundling type of the Twin Hub concept, as it can consolidate the flows of many terminals and only uses trains that move from terminal to terminal (i.e. trunk trains) instead of using shuttle trains that move between the hub and a terminal. Enlarging the service area of Rotterdam and Antwerp in a complementary way, and incorporating flows from smaller seaports are the source of the expected performance improvement compared to using single (per seaport) hub-and-spoke networks. The Twin Hub Network has two hubs, located in the gravity point of the Twin Hub flows: Antwerp and Rotterdam. Each train in the Twin Hub Network rides between a maritime rail terminal and an inland rail terminal. On its way, it visits one of the hubs, where it can exchange load units with other Twin Hub trains either directly or through the hub's stack. It is important to note that trains never visit both hubs, thereby limiting the costs of complex bundling.

1.1.4 Design considerations for Twin Hub Networks

The cooperation in the Twin Hub Network is one of competitors: both the seaports and rail operators are required to cooperate with their direct competitors to achieve the expected performance improvements. The conceptual logic of the Twin Hub project mentioned in the previous section shows that creating the Twin Hub Network can be profitable for all parties involved. Whether or not such a network is profitable in practice depends on whether companies are willing to use the offered intermodal transport services to transport their containers. It is therefore crucial to the success of the project to design a Twin Hub Network that will transport these expected flows at a cost that is as low as possible, while still offering an acceptable quality of service. Before the network can be designed, there are three unknowns that need to be filled in:

- What are the expected container flows?
- How are the costs calculated?
- What factors determine the quality of service?

These questions will be answered below.

Determining the expected container flows

The Twin Hub project is designed to stimulate intermodal rail-road transportation in North-West Europe, making these services more competitive compared to road transport. This means that the actual flows between origins and destinations are not known beforehand: it is difficult to predict what portion of the current flows by road will be shifted towards rail transportation.

Two of the project partners in the Twin Hub project will independently implement and use models to calculate how large this shift will be: At the faculty of Civil Engineering of the TU Delft, the shift of flows between transport modes will be predicted based on a logit model, and the Free University of Brussels will use the LAMBIT-model. As input for these model, statistical data of 2010 will be gathered and used as a basis for predicting the future flows. An important input parameter for these models will be the service frequency that will be offered in the Twin Hub Network: higher frequency services will likely attract more flows.

The result of these analyses are several matrices with predictions of future origin/destination flows for different service frequencies.

Cost calculation

There are many cost components involved in operating an intermodal transport service, for example fuel costs, labour costs, depreciation of equipment, and track access charges. Calculating the cost of every single component is possible, but this level of accuracy is not required, and would slow down the cost calculations significantly. Therefore, it is sufficient to decompose these costs into four components:

1. A kilometre-coefficient for operating a train or truck service.
2. An hour-coefficient for operating a train or truck service.
3. A transshipment fee per container transferred at the hub.
4. A fixed cost for pre- and/or post-haulage of a container.

Kilometre-coefficient The kilometre-coefficient is an estimate of the variable costs incurred for driving a train of a certain length and weight. The coefficient is composed of a locomotive and wagon costs. Examples of costs covered by this coefficient are electricity costs, infrastructure costs, and variable maintenance costs. These costs are dependent on the country the train travels through, as each country has different infrastructure and energy charges. Furthermore, energy costs are also dependent on the speed the train travels, as well as the weight of the train. For trucks, the kilometre-coefficient is composed of a fixed cost per kilometre driven.

Hour-coefficient The hour-coefficient is an estimate of the costs incurred for using a train of a certain length for a certain period of time. Operators consider these costs to be fixed, as these costs are not dependent on whether or not the train is operational. Examples of costs covered by this coefficient are labour costs, fixed maintenance costs, and depreciation of the equipment used (trains, wagons, containers, etcetera). These

costs are not incurred only when the train is driving, but for all operational time. The hour-coefficient is therefore incurred the whole year round, except for the weeks that regular maintenance is performed. The hour-costs of using a train to drive a certain train service is consequently not calculated for the hours that the train is actually driving, but for all the hours in which a train cannot be used for other train services. Factors that contribute to this additional time are buffering times, waiting times at the hub, and the fixed time window in which a train can drive – freight trains need to travel predominantly at night (the so-called night-leap), as they have to share the train-network with passenger trains. This means that, when the destination cannot be reached in one day, the train often has to wait until the next night (for most routes) before it can continue its journey. The difference between the operational round-trip time and the economical round-trip time can be significant, which means that the economical round-trip time must be used as a basis for calculating the costs. For trucks, the hour-coefficient is composed of a fixed cost per hour of truck usage.

Transshipment fee The transshipment fee covers all costs for the transshipment of one container from one train or truck to another. This fee covers all costs incurred at the hub, and is assumed to be a constant fee per container for simplification purposes.

Pre- and post-haulage costs The pre- and post-haulage costs covers the transport costs between the customer and the train terminal, the loading or unloading of that container at the customer, and the waiting times of the truck at the terminal and the customer. These costs are assumed to be equal and constant, and are incurred only by trains. Trucks do not incur these costs, as they can drive directly to or from the customer.

Schedule quality requirements

There are two important quality factors that must be addressed to ensure that the schedule is of an acceptable quality. One factor is the periodicity of the schedule. Figure 1.7 shows an example of such a periodic schedule, where the departures and arrivals must happen at the same departure days or arrival days, depart at the same time of day, and arrive at the same time of day. This way, a predictable and frequent schedule can be maintained for the customers of the transport companies, which is currently one of the biggest challenges in intermodal transport [24, 31].

The other factor is the way buffering and other delays are handled. Small delays and buffering times can be expected beforehand, and explicitly accounting for this is considered good practice, as this makes the schedule much more robust. For designing the schedule, this means that either waiting times must be explicitly planned, or to implicitly include this by calculating the train times using a slightly lower average speed than the trains can drive.

1.2 Designing Twin Hub Bundling Networks

In the Twin Hub Network, the bundling of flows from Antwerp and Rotterdam will lead to a substantial reduction in costs, while providing an acceptable quality of service. However, this reduction in costs relies on the design of the service network, and

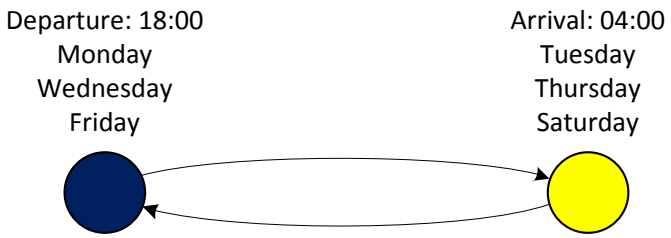

Figure 1.7: Example of a periodic schedule for one train service

therefore the quality of the bundling. Deciding upon which train services to organise, and when these train services should drive is no simple feat, as there are billions of options to choose from for all but the smallest networks.

The structure of the bundling operations has to reduce this complexity as much as possible. Therefore, instead of deciding what the exact train times should be for every train in the network, we use the concept of batches to create discrete intervals in which trains can visit the hub. These intervals are of a fixed length, and should be long enough to transfer the containers between all trains present in a batch. Naturally, there are restrictions on the number of trains that can exchange load units during a batch, as there are only a limited number of tracks available at a hub. Also, the capacity of the stack at the hub where the containers may be stored temporarily in between batches has some finite size which should not be exceeded in between batches.

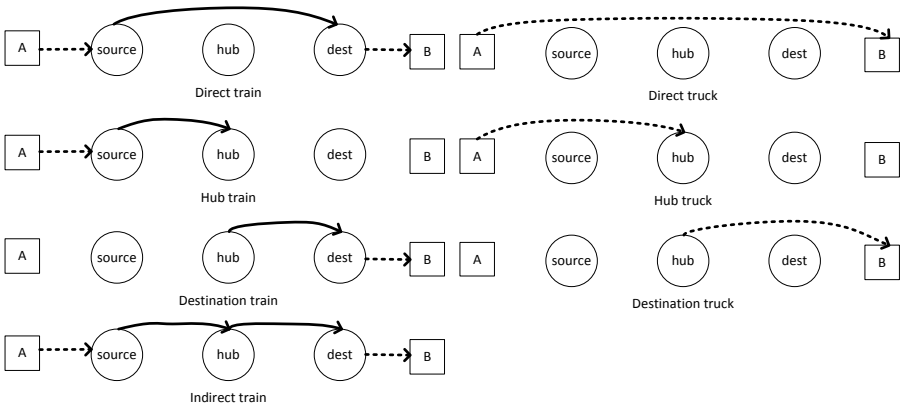

Figure 1.8: The types of train and truck services that can be created

While a Twin Hub network as defined by the Twin Hub project only contains indirect trains and may also include trains from the smaller seaports to the hub, the modelling of these networks may include other types of trains to aid in finding the optimal Twin Hub Bundling (THB) networks, even though these may not seem logical to transport companies. The resulting set of transport services does not only consist

of batches with trains that visit the hub, but may also contain direct trains, or trucks that drive the same routes as the trains. Figure 1.8 shows the seven different transport services that may be part of a service network, where the continuous line represents a train service, and a dotted line a truck service. The squares A and B represent the source and the destination of the container respectively. To transport a container from A to B, it can either use one of the train services (left column) or travel by truck (right column) instead. Whenever a container travels by train for part of its journey, it must be transported (pre-haulage) towards the nearest train terminal ('source') by truck first, if the train terminal is not directly next to A, which may be the case for some seaports. The same holds for the post-haulage, where trucks are used to transport the container from the destination terminal ('dest') to its final destination B. A container can also participate in the complex bundling process, in which it can take any combination of train or truck services when it is transported from A to B.

Choosing which train and truck services to offer, and which containers should drive on which train or truck is a complex problem, which is very hard to solve by hand. For each container to be transported, it must be decided which combination of transport options is optimal, which again depends on the other container flows to be transported. Opportunities for reducing costs in one place by combining certain flows on one train may cause higher costs to be incurred elsewhere, as a better combination could have been made when bundling the flows in a different way.

With many flows to combine on trains, the number of possible train and truck schedules is huge. Attempts to design hub-and-spoke networks for a limited number of terminals for the Rotterdam were successful in reaching the goal of that project, but proved to be very time consuming, even for a fraction of the number of terminals the Twin Hub project wants to connect. Even for the most advanced computers, capable of assessing the quality of possible schedules at very high speeds, simply examining all possible schedules would be virtually impossible. The question is whether there exists an efficient way of finding the optimal bundling network using a computer, which is something that can be proven mathematically. Actually finding the optimal bundling network may therefore prove to be impossible, though an algorithm might still be devised that produces good-quality bundling networks in a reasonable amount of computation time if this is the case.

When an algorithm is created that automates the creation of THB networks, this algorithm can be used in the Twin Hub project to evaluate certain design choices when implementing these networks. For example, the influence of offering a higher service frequency or allowing more trains to meet at the hub in a batch might be evaluated, which would be extremely time-consuming or even impossible to do by hand.

The evaluation of the resulting THB networks produced with different settings is, consequently, also an important feature that needs to be implemented. Therefore, a bundling tool should be developed that facilitates this evaluation procedure, allowing the partners in the Twin Hub project to choose the most promising THB network to implement in the pilot that will be organised in 2013. Initially, this network will consist of only the most promising batch, though when the pilot proves to be successful, the same tool may be used to find additional promising networks to expand the Twin Hub network.

1.3 Bundling tool

The purpose of creating the bundling tool is to design the most promising set of transport services to move all flows in the projected matrix of flows between the seaport rail terminals in Belgium and The Netherlands, and the inland terminals throughout North-West Europe and further. The set of transport services includes services belonging to Twin Hub Bundling (THB) networks. The tool is useful for identifying best Twin Hub Networks on the short to long term. It will be used to evaluate the consequences of choosing a certain service frequency on the resulting bundling network. The challenge is to design THB-networks with the minimum cost possible, while achieving a set service level.

Beside designing promising THB-networks, the bundling tool has several other requirements to allow it to be used in the Twin Hub project. These requirements are split in three types: performance requirements, control requirements, and user interface requirements. Each of these will be specified below. Note that a more elaborate requirements document is included in Appendix C.

1.3.1 Performance requirements

The performance requirements concern the algorithm that finds the THB networks. When the algorithm that is included does not find good-quality THB networks within a reasonable amount of time, the bundling tool cannot be used to find and evaluate promising bundling networks. It is therefore essential that the following requirements are met:

1. The algorithm must be able to compute a reasonable (i.e. non-trivial) solution to a "real-world sized problem", i.e. a problem with 20 sources, 50 destinations, and 60 batches, within 24 hours.

2. The algorithm must be able to meet the above requirement using a machine with a modern quad-core processor and 8GB of RAM.

1.3.2 Control requirements

The bundling tool needs to design a THB-network that transports the predicted flows for a certain service level. This means that each parameters of this problem instance must be specified in order to compute the optimal THB-network. These parameters are:

1. **Origin/destination flows**: The flows between terminals in the network;

2. **Service frequency**: The service frequency that should be achieved by the THB-network that is designed;

3. **Network distances**: The distances between terminals and between terminals and the hub;

4. **Costs**: The cost parameters as specified in Section 1.1.4;

5. **Train capacity**: The capacity of each train in the network;

6. **Hub capacity**: The number of trains that can exchange load units at the hub in one batch.

Additionally, parameters needed for the operation of the algorithms should be controllable as well. These include:

7. **Runtime**: The maximum runtime of the algorithm;
8. **Maximum number of batches**: Some algorithms require a maximum number of batches to be used in a solution as an input to function properly;
9. **Other algorithm-related parameters**: Algorithms may provide other settings that must be controlled in order for the algorithms to operate. Examples are the amount of memory the algorithm can use, and parameters that control refinements of the search strategy. All these values must have a default option, so no in-depth knowledge of the algorithms is required.

1.3.3 User interface requirements

The user interface (UI) determines not only the look and feel of the tool, but is also an important factor in the usability of the tool. A good UI allows the users of the tool to reach their goal quickly and intuitively, and increases their productivity. Therefore, it is important to specify the requirements for the UI that is to be developed. These requirements are the following:

1. The user should be able to specify a problem in the tool. That is, the user must be presented with a means of entering the control parameters specified in Section 1.3.2, so a complete specification of a problem with all its parameters is made;
2. The user should be able to start the optimisation algorithm to find the solution to the specified problem;
3. The user should be able to view the following aspects about the Twin Hub Bundling (THB) networks generated by the algorithm:
    a) The THB-networks themselves: which train and truck services are created, and containers from/to which terminals are transported on each train or truck;
    b) Statistics about the generated THB-networks: ‘cost’, ‘cost per LU’, and ‘modal split’ about the whole solution, and per batch, train, and truck, where applicable.

1.4 Thesis focus and outline

The goal of this thesis is to develop the bundling tool for the Twin Hub project, with which promising Twin Hub Bundling (THB) networks can be identified based on projected container flows between seaports and inland terminals. In the project, the tool will be used to evaluate different scenarios and parameter settings to find the most promising THB network.

1.4.1 Research questions

In order to find promising THB networks, an algorithm should be included in the bundling tool that solves the THB problem. As there is no algorithm currently available for this problem in the project, this algorithm will have to be developed and implemented.

Developing an algorithm requires insight into the problem and its structure, as different types of algorithms should be used to solve problems with different properties. One of the properties that is especially important for choosing which kind of algorithm can be used to solve the problem is the theoretical complexity of the problem. When a problem has a high complexity, it is very improbable that an efficient algorithm can be found that can solve the THB problem, and other types of algorithms should be used to achieve a good-quality solution within a reasonable amount of time than when the complexity of the problem is low.

Existing problems in transport literature may be useful in investigating which kind of algorithms are used to solve similar problems. It may also be possible that an algorithm used to solve a similar problem can be modified so that it can be applied to solve the THB problem. When the algorithms applied in the existing literature cannot be used, an algorithm has to be designed from scratch.

After the algorithm has been developed and implemented, an experimental analysis should be performed to characterise the performance of the implemented algorithm to check whether it meets the performance requirements as stated in Section 1.3.1.

To reach the goal of developing a bundling tool and in particular the algorithm that has to be included, we formulate the following research questions:

1. What existing research has been done on problems similar to the THB problem?
2. Can we characterise the complexity of the THB problem?
3. If similar problems exist in literature, can we apply the algorithms that solve this problem to solve the THB problem? If not, can we design an algorithm that solves this problem?
4. Can we characterise the performance of the algorithm(s)?

The resulting algorithm(s) will then be incorporated in the bundling tool, which will be developed to provide an easy to use interface for running the algorithm and to view the algorithm's results.

1.4.2 Contributions

In addition to the goal set by the Twin Hub project for the development of the tool, these research questions will also lead to contributions to the field of transport research. The most important contributions are:

1. The characteristics of the THB-problem will be compared with existing problems in intermodal transportation, and the innovative features of our modelling will be highlighted.
2. A new model will be created, incorporating the new features.

3. The complexity of the THB problem will be analysed.
4. Two algorithms for solving the THB problem will be created.
5. A computational analysis of the two algorithms will be performed, comparing their performance and analysing the characteristics of these algorithms.

These contributions will lead to an advance in the field of intermodal transport research, both in terms of the modelling of intermodal transport networks and the creation of algorithms for solving these models.

1.4.3 Outline

The remainder of this thesis will be structured as follows: first, a literature survey will be performed in Chapter 2 to investigate what existing research has been done. Then, a thorough analysis of the Twin Hub Bundling (THB) problem will be performed in Chapter 3, providing a definition of the problem and an analysis of its complexity. The design of algorithms for solving the THB problem will be provided in Chapter 4, after which a computational analysis of these algorithms will be performed in Chapter 5. In Chapter 6, the design of the bundling tool will be discussed. Finally, in Chapter 7, the contributions of this thesis will be presented and a conclusion will be drawn answering the research questions. Suggestions for future work will be presented as well.

Chapter 2

Literature Survey

Intermodal transportation research is a large research field in which many Operations Research models and methods have been applied to optimize the transportation planning and operations. In the past three decades, a significant body of research in intermodal freight transport has been created, tackling many of the optimization problems that exist in this field. However, the sheer size of a typical transportation model has prevented the creation of an integrated approach that deals with all problems in designing an intermodal freight network simultaneously. Research therefore usually focuses on a small aspect of the problem, leading to many different problem definitions and, consequently, fragmentation.

The problem to be solved in this thesis is the Twin Hub Bundling (THB) problem, which consists of designing both train and truck services and bundling services to transport a predetermined amount of flows through the network between a group of port terminals and a group of inland terminals. Each train or truck service can use one of two hubs to exchange load units with other trains or trucks, depending on the inland terminal visited by the service. Each transport service visiting the hub is assigned to a batch. Load units can only be exchanged between transports visiting the hub in the same batch, directly, or through the stack at the hub, where the load units become available for later batches to pick up. The goal is to minimise the total costs involved in transporting all origin/destination flows, including the operations at the hub, and pre- and post haulage.

2.1 Surveys

To investigate whether problems similar to the THB problem have been described in literature, several surveys of transportation literature have been consulted. An influential survey paper by Crainic and Laporte [9] has identified and classified many of the issues in freight transportation in general. They structured the research field according to the three classical decision-making levels: the strategical, tactical, and operational level, concerning long-term, medium-term, and short-term questions respectively. The THB Problem is a problem at the tactical planning level, and can be classified as a service network design problem for intermodal transportation. The main research issues identified for this category of problems by Crainic and Laporte are the following:

1. Service network design, in which the routes and frequencies for the trains are determined.

2. Traffic distribution, in which the routing specifications for each pair of begin- and end terminals and the operations performed at the terminals are specified.

3. Transport node policies that specify for each terminal the type of consolidation to perform, e.g. for blocking trains, the number of blocks and the allocation of blocks to trains is performed.

4. Empty balancing, in which the empty containers are redistributed through the system.

5. Crew and motive power scheduling, which describes how to allocate the resources required in preparation for the next planning period.

These tactical planning issues must be dealt with concurrently or simultaneously, with the latter strategy yielding larger, and consequently harder to solve problems that may yield better results overall. In our problem, we focus on *traffic distribution* and *transport node policies*. In the case of the THB-problem, the transport node policies are replaced by bundling policies, which are somewhat more general in the sense that the policy does not apply to each transport node individually, but to the entire network. The routes and desired service frequencies for each origin-destination pair are part of the input, while empty balancing and crew and motive power scheduling are dealt with separately.

An extension to this model was proposed by Macharis and Bontekoning [28] in their survey of the opportunities for Operations Research in intermodal freight transport research, which addresses the issue that not all decision steps are made by the same actor. Caris et al. [4] have updated this survey to include the OR literature up to 2008. The authors make a distinction between drayage operators, terminal operators, network operators, and intermodal operators. Within each combination of decision maker and time horizon, a number of planning problems are identified and relevant papers are mentioned. The problem solved in this thesis fits best in the category of "Configuration consolidation network". While the decision to use a hub-and-spoke network has already been made, establishing a train service schedule, including determining the frequencies of different types of trains between terminals, falls into this category.

Cordeau et al. [7] have surveyed optimisation models for train routing and scheduling. They distinguish different types of issues, such as train routing, freight car management, and train dispatching, though they do not present a clear taxonomy of these models. This survey is relevant nonetheless, as the papers about train schedule optimisation that have been surveyed do not appear in the other surveys.

2.2 Literature search

A literature search guided by the surveys mentioned above was performed to identify papers solving similar problems to the one solved in this thesis. First, the relevant papers found in the survey by Caris et al. [4] are discussed, after which the papers

found in Cordeau et al. [7] are discussed. Last, two other relevant papers not included in one of these surveys are presented.

2.2.1 Papers identified in the survey by Caris et al.

The first work identified in the survey by Caris et al. [4] was a paper by Janič et al. [20], who describe the internal and external cost components of a bundling network. However, in this thesis we are predominantly interested in creating an inherently profitable intermodal hub and spoke network by identifying the most promising networks. External costs are, therefore, not taken into account at this point in the Twin-Hub Network project. However, these costs will be taken into account once the total benefits of this project are calculated at the end of the Twin Hub project.

The two papers by Newman and Yano [29, 30] also mentioned in the survey by Caris et al. [4] are very similar, and describe an approach to centralized and decentralized scheduling of direct and indirect trains. The problem described in these papers is quite similar to the problem dealt with in this thesis. It involves the scheduling of three types of trains: trains from the source terminals to a central point consisting of one or more hubs, trains from the hub(s) to their destination, and trains that run directly from a source terminal to a destination terminal. Trains that run from a source to a destination terminal via the hub ("indirect trains") are not considered in this problem. The planning approach is also different, as the goal of the planning is not to create a periodical schedule like in the THB-problem, but a non-recurring schedule where the number of containers may vary across the different time periods. This is done by considering a limited number of time periods, denoting a planning period of about two weeks. Container flows through the network all have a certain storage cost associated with them, and have to be delivered before a certain time period. Terminal operations are also modelled, and transfer costs are incurred for each container passing through the hub, as only services between a terminal and the hub are considered. This is where our approach differs as well, as in the THB problem we also consider indirect trains, where transfer costs are not incurred for containers that remain on the train when the train visits the hub. Transfer costs are also assumed to include the storage costs, as the planning horizon is much shorter due to the periodical nature of our model.

The solution methods described by Newman and Yano include a centralized scheduling approach using integer programming and a decentralized approach based on a decomposition of the planning problem in three planning steps: first, the direct trains are scheduled, then the trains to the hub (from the origin), and finally the trains to the destination (from the hub). The individual problems are then solved independently, and combined to form the solution for the main problem. Additionally, two simple heuristics are presented, but yield solutions that are 15% worse than the decomposition algorithm. The computational results are then evaluated, and the decentralized approach yields comparable solutions to the integer programming approach, while the runtime is significantly shorter. However, due to the differences between our model and that of Newman and Yano, the decentralized approach is not applicable to our problem, as including indirect trains means that the problem cannot be decomposed in a to-hub and from-hub part anymore, but should be considered simultaneously.

2.2.2 Papers identified in the survey by Cordeau et al.

Cordeau et al. [7] have surveyed optimization models for train routing and scheduling. Many papers that have been surveyed focus on operational problems, such as freight car management or train dispatching models. The papers surveyed that are concerned with tactical planning are dealing with different problems than the problem to be solved in this thesis. Crainic et al. [8], for example, examine the routing of container flows and trains through a network based on line bundling and different traffic classes (i.e. priorities), and therefore focusses more on routing through the physical network, while in the THB problem, trains can only take a specific set of services, either through direct services or services visiting the hub. Crainic and Rousseau [10] present an algorithm for optimizing train schedules as well as traffic distribution. However, this algorithm cannot be applied to the THB problem, as the quality of the produced train schedules cannot be controlled, as well as many other elements (e.g. hub processes). Haghani [18] presents an elaborate system attempting to integrate routing, train makeup (blocking), train frequencies, and empty car distribution. Their model focusses primarily on the design of train routes through the network and (empty) car distribution, which are both issues that are not present in the THB problem. Additionally, the algorithm produces schedules that do not incorporate hub-and-spoke networks using batches, are not suitable for incorporating night-jumps, are not periodical, and which are therefore unsuitable to produce a solution to the THB-problem.

2.2.3 Papers found in other sources

Groothedde [14] has done research on collaborative logistics in hub networks, in which many forms of collaboration in transportation networks are examined. The dissertation contains useful background information, as well as a good review of literature on the modelling of decisions in logistics [14, ch.4]. The problem described in this dissertation focuses on the hub location problem and service network design. It also incorporates an actor model that describes how participants might use the resulting network to calculate the potential benefit. The hub location and the response of actors to the resulting network are clearly outside the scope of this thesis. The service network design is based on a hub-network, where multiple hubs are located at a large distance from each other. A shipment travelling from A to B is modelled to have two options: it can travel directly from A to B on a train, or it can be transported through the hub-network. In this network, the shipment is first transported to the nearest hub on the ‘spoke’ running from the terminal to the hub. From there, it is bundled with other shipments towards the hub nearest to its destination. Then, it is again transported on the ‘spoke’ from the other hub to its destination terminal. It is not modelled how these trains actually drive through the network: the modelling only includes flows between connections, not the trains and/or trucks that transport these flows. While economies of scale are considered in this model, they are modelled separately as a discount on the price of using a particular connection between a terminal and a hub, or between hubs. The part that we are particularly interested in – finding THB networks with indirect trains in a batch structure – is not modelled. Extracting these networks from a solution to the model of Groothedde is difficult, as the train and truck services would still need to be designed, which is non-trivial. Finally, each container in the network

that uses the hub-and-spoke network option incurs transfer costs (and other costs associated with the use of the hub) for each hub visited. This modelling is therefore unsuitable for solving the THB problem.

Grünert and Sebastian [15] have developed an algorithm for planning tasks in an intermodal hub-and-spoke network using aeroplanes and trucks to deliver mail and packages under time constraints. Even though the domain is slightly different, the problems encountered are similar to the problems in intermodal train transport. The air transportation problem is, for example, very similar to the train service design problem, as direct flights and hub flights are considered when routing mail and packages through the transportation system. Constraints are placed on the capacity of the airports, and packages that are routed by air are delivered by truck instead if the delivery times allow this. To solve the authors' problem, an integrated model is created in which the relations between air and ground traffic are explicitly modelled. While the problem solved in this paper has many elements in common with the THB problem, such as the choice between direct services and services that visit a hub, and an alternative way of transport (trucks), there are a number of key differences which makes this modelling unsuitable for solving the THB-problem. For example, the modelling only considers 'to-hub' and 'from-hub' flights that are scheduled individually. This means that there is no concept of 'indirect flights' nor batches of aircraft that are serviced simultaneously. This means that every item transported incurs the same costs when transported on a particular flight. Finally, the model presented in the paper is centred around delivery times that must be met, instead of minimising the costs while providing a certain (fixed) quality of service. The decision whether or not to use trucks is not driven by costs like in the THB problem, but by delivery time constraints. Trucks are only used to transport an item when the delivery time constraints when the delivery time window of that item allows it. Otherwise, the item is transported by air.

2.3 Discussion

Transportation research is a large research field, where a sizeable amount of research has been done on a wide number of optimization questions. However, even with the current volume of research that has been done, no literature has been found solving a problem similar to the one solved in this thesis. Partly, this is due to the innovative nature of the Twin Hub project. Important concepts that have been introduced in the THB problem that have not been modelled in existing intermodal transport literature are the simultaneous consideration of both road- and rail transport alternatives, the organisation of trains in 'batches' that meet at a central hub simultaneously to exchange containers, and the modelling of so-called indirect trains, which start and end at a seaport terminal and an inland terminal while visiting the hub in a batch, enabling some containers on the train to be exchanged with other trains, while other containers do not partake in the sorting process, saving costs.

Another reason why no suitable literature could be found is that research in the field of transportation planning and scheduling is fragmented. As there are so many variables involved when planning or scheduling transport services, it is impossible to incorporate all variables in a single model, as this model would be too large and complicated to design and solve. Some models have incorporated many different variables,

such as the model created by Groothedde [14], but a universal modelling for solving planning or scheduling problems is still lacking.

When looking at the models in the papers discussed in this chapter, the model and algorithms proposed by Newman and Yano [29] are the most relevant for our problem, as this problem has a similar domain and optimization goals. In this thesis, the model and solution method of this paper will therefore be used as a basis for creating a solution to the Twin-Hub bundling problem. To investigate whether this is possible, we will first define the problem and investigate its characteristics in the next chapter, before the algorithm to solve the THB problem is presented in Chapter 4.

Chapter 3

Problem Analysis

The literature survey (Section 2) has shown that the Twin Hub Bundling (THB) problem is not solved in existing transport literature. New concepts that are introduced in the THB problem are indirect trains, the organisation of intermodal transport services in batches, and the simultaneous scheduling of trains and trucks. In this chapter, a new problem definition incorporating these new concepts will be presented, after which the complexity of this problem will be analysed.

3.1 Problem setting

The Twin Hub Bundling (THB) problem faced by the Twin Hub project is an optimisation problem in the transportation domain. The setting is a transportation network with two groups of terminals: port terminals and inland terminals. A rail network and a road network connect these groups of terminals, and on these networks trains and trucks can drive respectively. Each port terminal is connected with each inland terminal, so a transport service can be set up between each combination of inland and port terminals. These transport services can transport intermodal load units (i.e. containers), which can travel on both the train and the road network.

Between each port terminal and inland terminal, there may be a demand for a transport service in the form of intermodal load units having to be transported to another terminal. This demand must be met by the train and truck services in the network in a way that is as cheap as possible.

Figure 3.1 shows a small instance of the THB problem with three port terminals (dark blue) and four inland terminals (red, yellow, green, light blue). At each port terminal, there are flows of different sizes to be transported to each inland terminal using trains and trucks. In this problem, we assume that the required service frequency is 1. With higher service frequencies, we would have to divide the available flows by the service frequency, yielding smaller flows per time period. Now, the flows shown in the figure are the flows that have to be transported using trains with a capacity of 60 load units, or using trucks with a capacity of 1.

If only direct train services would be used, 12 train services would be required each time period to transport all the flows. Fortunately, there are alternative, cheaper options in the THB problem for the flows to be transported. Flows can be bundled at one of two hubs: Antwerp and Rotterdam. The choice between each of these terminals

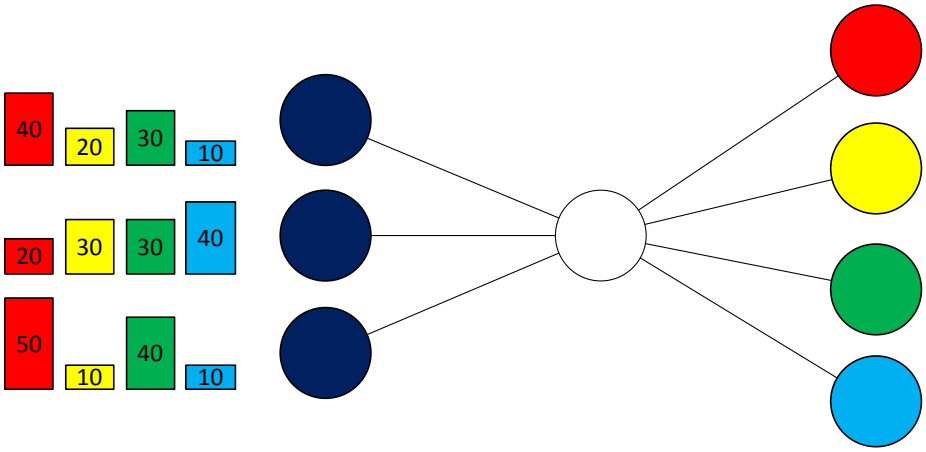

Figure 3.1: The THB problem to be solved

is made based on the inland terminal the train attends: load units destined for the North-East corridor use the hub Rotterdam, and load units for the South-West corridor use Antwerp as a hub. In practice, this means that the inland terminals can be divided in two groups: the inland terminals serviced by Rotterdam, and the terminals serviced by Antwerp. As these groups do not interact, the problem instance can be split in two subproblems, each with only hub. Combining the solutions to these two problems yields the solution to the original problem.

Furthermore, the resulting subproblem can be split again in distance relations: each inland terminal belongs either to an A/B, A/C, or A/D relation, depending on how many days it takes for a train to reach that inland terminal. An A/B relation means that the round-trip time for that terminal is one day, for an A/C terminal this is two days, and for an A/D terminal three (or more) days. Trains with different distance relations do not mix, meaning that a train to an A/B inland terminal cannot exchange load units with a train to an A/C inland terminal. This means that the problem can also be subdivided based on these categories.

Finally, it is acceptable to solve only one ‘direction’ at a time, so to solve the problem of transporting containers from the port terminals to the inland terminals first, and then solve the problem of transporting containers from the inland terminals to the ports. Solving these problems can be done using the same algorithm, as the two groups of terminals can be switched to compute the ‘return’ direction, dividing the problem in half again.

The problem shown in the figure is, therefore, one of these subproblems, where only one hub can be used to combine flows, with inland terminals with one distance relation, and solved in one direction. When solving this problem, direct trains and trucks are considered, as well as five types of hub-and-spoke trains and trucks. Trains can drive from a terminal to the hub (‘hub trains’), from the hub to a terminal (‘destination trains’), or from a terminal to another terminal, while visiting the hub in between (‘indirect trains’). Trucks can also visit the hub, but can only drive from a terminal to

the hub ('hub trucks'), or from the hub to a terminal ('destination trucks'), as scheduling indirect trucks (having a capacity of 1 load unit) would not be useful, since this would be equivalent to scheduling a hub truck and a destination truck.

These transport options also need to be considered by the algorithm, as these may yield a better solution to the THB problem. Services visiting the hub need to be assigned a batch number to identify when these services visit the hub. This is important to determine the order in which trains are serviced, and to keep the exchange with the stack consistent: trains can only retrieve containers from the stack if these containers are placed on the stack by a train with a lower or equal batch number. This stack has a limited capacity, but this limit is usually chosen large enough to ensure this is not a limiting factor for the solution. Also, the number of trains that can visit the hub simultaneously is limited, due to the number of tracks available at the hub for trains to exchange load units. The amount of trucks that can visit in a batch is not limited, as trucks can place their load units on the stack in between batches.

When solving the THB-problem, the algorithm must ensure that a consistent schedule is created. This means that all containers must be transported, and that the transfers between trains are consistent. The periodicity is ensured by computing only one time period, which can then be repeated as many times as necessary. To meet the requirement that the schedules conform to night-jump times, we implicitly define a time window in which trains are allowed to drive by limiting the number of hours a day a train can drive. The distance relation then determines what the economical round-trip time is. As only half a round trip is scheduled, this 'half economical round-trip time' must be specified too. This is always a multiple of 12 hours (half a day), and determines how many hours the train is occupied by this train schedule. Other factors determining the costs of a transport service are the number of containers carried, the distance travelled, and the number of containers that are exchanged at the hub. Using these parameters, the costs of a train or truck service can be calculated. The types of costs are described in more detail in Section 1.1.4.

Returning to our example, we now know what problem to solve: the transport options to choose from, the optimisation criteria, and the requirements the solution must conform to. Solving even this small problem is not straight-forward, as there are many possible schedules that transport all flows and conform to all other requirements to the schedule.

Figure 3.2 shows a solution to the problem in Figure 3.1 containing two batches with three trains each. In the first (top) batch, the three trains are completely filled with containers, so a 100% utilisation rate is realised. For the second batch, this utilisation rate is somewhat lower. The cost calculation shown at the batches is based on a fixed train cost of 2000, variable costs per container of 50 (25 to the hub, 25 to the destination), and a transfer cost of 25. The different types of lines show the indirect train routes and determine the number of transfers for each train. While it may seem that this configuration is the optimal configuration – we have a 100% utilisation rate for the first batch with the second batch transporting the remainder of the containers – this 'greedy' approach of forming batches with the highest utilisation rate possible is not always the best.

Figure 3.3 shows a different solution to the problem, with a reduced train utilisation in the first batch: 10 containers towards the blue terminal are left at the hub for the second batch to pick up. Another difference is that only two trains participate in the

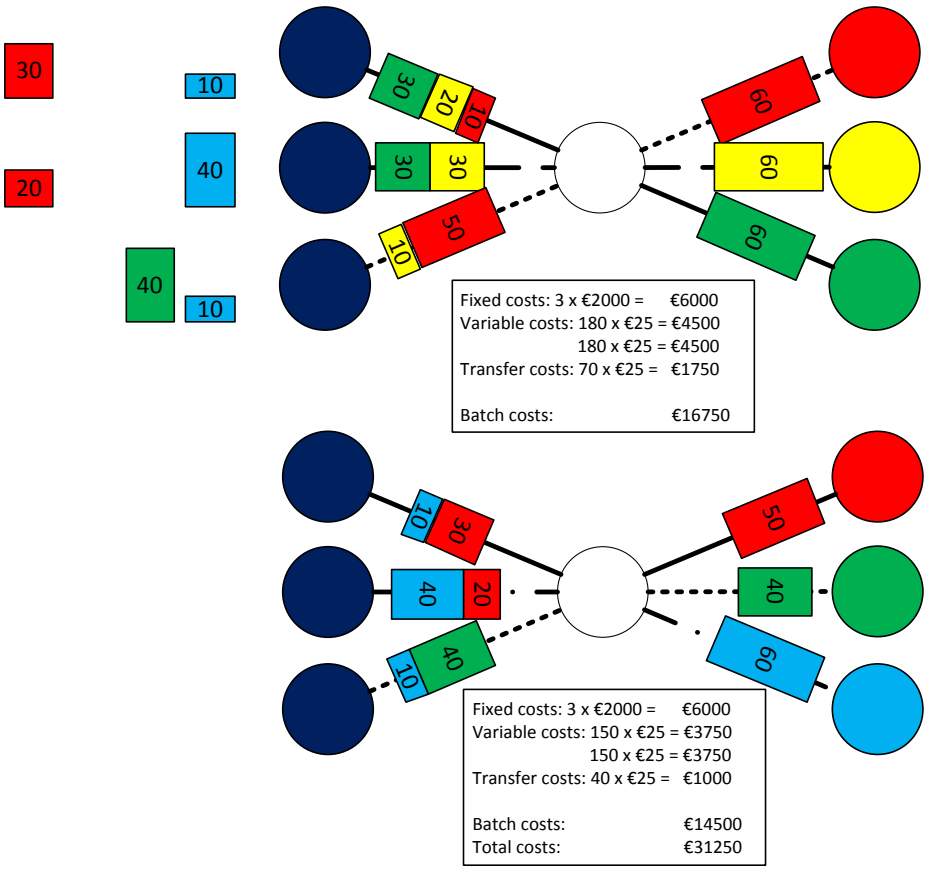

Figure 3.2: A suboptimal solution to the THB problem in Figure 3.1

second batch, and a direct train is used to transport the 50 red containers at the third port terminal. These minor differences lead to a cost savings of 250 in this fictional problem by decreasing the amount of transaction costs incurred. In real-world networks, with a realistic calculation of costs and larger networks, the cost savings may add up to 10 to 15% of the total costs.

Finding good-quality schedules is therefore key to creating profitable THB networks. Saving just a few per cent in costs may mean a savings of tens of thousands of euros for larger networks. Therefore, an accurate problem formulation is required to create an algorithm that solves the THB problem.

3.2 Formal problem definition

The formal problem definition describes all aspects of the Twin Hub Bundling (THB) problem in detail. Most of the formulation follows directly from the description given in Section 3.1. However, two aspects of the problem definition need more explanation: the structure of the train network and the calculation of the costs. These are described first, before the formal problem definition is given.

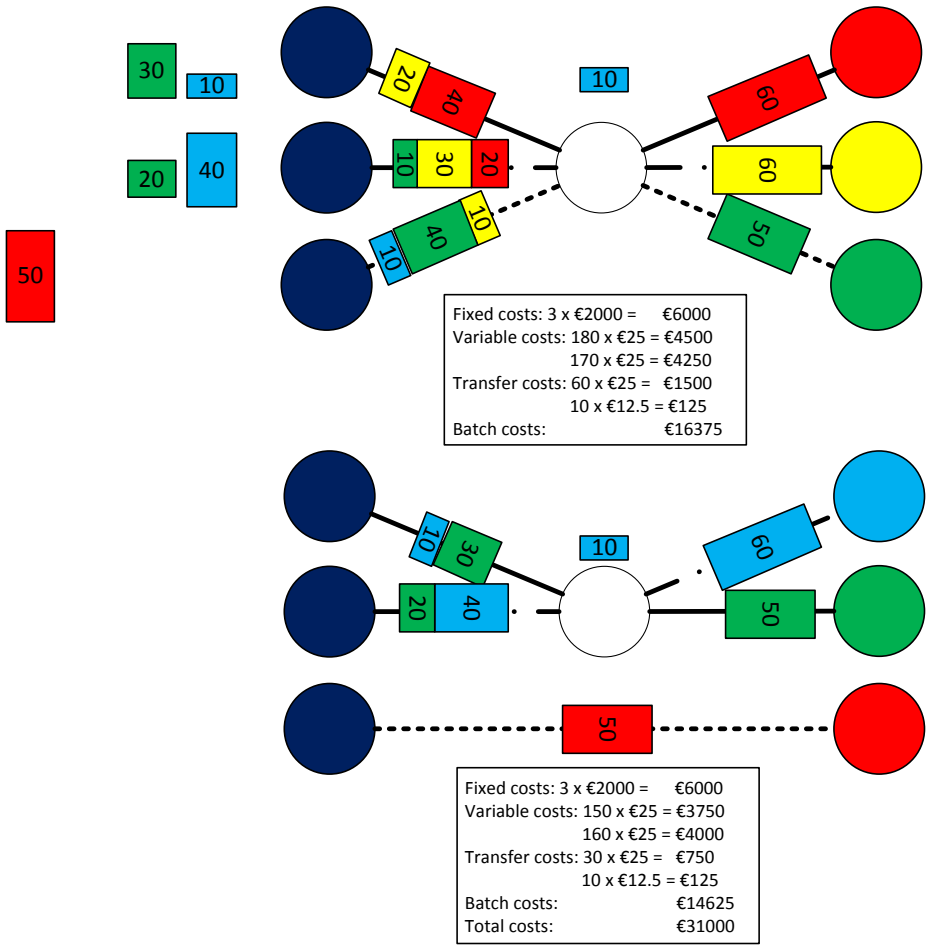

Figure 3.3: The optimal solution to the THB problem in Figure 3.1

The structure of the train network is modelled as a graph G, with the nodes V representing the terminals in the problem and the edges E describing the connections between nodes. We distinguish three types of nodes: the port terminal nodes P, inland terminal nodes I and the hub h. Each inland terminal and port terminal are connected using direct connections E^{direct}, and the terminals in both groups are connected to the hub by edges E^{hub}. All edges are labelled with the distance $dist(e)$ and the travelling time $time(e)$ between terminals. For each pair of nodes, the amount of containers to be transported from one node to the other is specified.

The total cost of running a train t can be calculated by determining the total distance $dist(t)$ travelled by the train, the time $time(t)$ it takes to travel that distance, the number of containers L transported by the train, and whether or not the train attends the hub. Pre- and post-haulage costs also need to be taken into account if applicable. These costs are then added together to compute the total costs of that train. Truck costs are calculated in a similar manner. Adding the costs for all trains and trucks in the network yields the total network costs, which need to be minimized.

The following definition will be used in this chapter to investigate the characteristics of the problem. This is a restriction of the original problem, meaning that only the variables which are relevant to the problem's complexity are shown. A full definition can be found in Appendix A.

TWIN-HUB BUNDLING (THB):
Given:

- A graph $G = (V,E)$ constructed as mentioned above, representing the network.
- A list of possible batches B with $|B| \in \mathbb{N}^*$.
- The flow of containers F_{ij} between terminals i and j with either $i \in I, j \in P$ or $i \in P, j \in I$.
- The maximum number of trains that can exchange load units at the hub at the same time H, i.e. the capacity of one batch $b \in B$.
- The fixed cost per hour for using a train C_{time}^{fixed}. Note that use is defined as being part of the economical round-trip time of the train, not the time the train is actually driving.
- The fixed cost per kilometre for driving a train C_{km}^{fixed}.
- The variable cost per load unit per hour for using a train $C_{time}^{variable}$.
- The variable cost per load unit per kilometre for driving a train $C_{km}^{variable}$.
- The total cost per hour for using a truck C_{time}^{truck}.
- The total cost per kilometre for using a truck C_{km}^{truck}.
- The hub transfer costs of one load unit $C^{transfer}$.
- The costs for pre- or post-haulage for one load unit C^{pph}.
- The maximum capacity W of a train.
- The maximum stack size M.

How can each flow F_{ij} be allocated to a set of trains T and trucks R that may participate in one of the batches in B, while minimizing the following cost function:

$$\begin{aligned} Z &= \sum_{t \in T} (cost^{fixed}(t) + cost^{variable\_direct}(t) + cost^{variable\_hub}(t) + cost^{variable\_dest}(t)) \\ &+ \sum_{r \in R} (cost^{total}(r)) \end{aligned}$$

Where:

- v_t^{start} = the start terminal of transport t
- v_t^{end} = the end terminal of transport t
- $L_{v_1,v_2,t}$ = load transported from v_1 to v_2 on transport t with $v_1, v_2 \in V \setminus \{h\}$.
- $L_{v_1,h,t} = \sum_{v_2 \in V \setminus \{h\}, v_1} L_{v_1,v_2,t}$
- $L_{h,v_2,t} = \sum_{v_1 \in V \setminus \{h\}, v_2} L_{v_1,v_2,t}$
- $cost^{fixed}(t) = C_{time}^{fixed} * time(t) + C_{km}^{fixed} * dist(t)$
- $cost^{variable\_direct}(t) = L_{v_t^{start},v_t^{end},t}(C_{time}^{variable} * time(t) + C_{km}^{variable} * dist(t) + 2 * C^{pph})$
- $cost^{variable\_hub}(t) = L_{v_t^{start},h,t}(C_{time}^{variable} * time(e_{v_t^{start},h}) + C_{km}^{variable} * dist(e_{v_t^{start},h}) + C^{pph} + 0.5 * C^{transfer})$
- $cost^{variable\_dest}(t) = L_{h,v_t^{end},t}(C_{time}^{variable} * time(e_{h,v_t^{end}}) + C_{km}^{variable} * dist(e_{h,v_t^{end}}) + C^{pph} + 0.5 * C^{transfer}))$
- $cost^{total}(r) = C_{time}^{truck} * time(r) + C_{km}^{truck} * dist(r) + 0.5 * C^{transfer}(L_{v_1,h,r} + L_{h,v_2,r})$

Under the following constraints:

1. Trains and trucks can only exchange load units with each other if they belong to the same batch $b \in B$.
2. Batches have an order in which they are processed, allowing for interaction of trains and trucks with the stack so that containers can be temporarily stored between batches b_i and b_j provided that $i \leq j$.
3. The desired frequency S is achieved on all connection-pairs in the system where $F_{ij} > 0$.
4. The resulting schedule is periodical.
5. After each batch, the number of containers stored on the stack is at most M.

3.3 Problem Characteristics

The formal problem definition of the Twin Hub Bundling (THB) problem describes a problem that is different from the problems found in existing literature. This means that we must investigate the problem's characteristics before an algorithm can be developed. Most planning and scheduling optimisation problems found in literature are NP-hard, and the problem described by Newman and Yano [29] which served as a basis for the formulation of the THB problem is no exception. This means that the THB-problem is probably also an NP-Hard optimisation problem.

In this section, we will prove that the THB-problem is an NP-Hard optimization problem by proving that the decision variant of the THB-problem (THB-dec) is NP-Complete. When the decision variant of an optimisation problem is NP-Complete, the optimisation problem itself is NP-Hard. First, we will present the THB-dec problem, after which we show this problem to be in *NP*. Then, an NP-Completeness (NPC) proof for THB-dec will be presented.

3.3.1 Decision variant of THB

Below, we present the decision variant of the THB problem, which will be used in the NP-Completeness proofs.

THB-dec:
"Given an instance $P = (B,C,F,G,H,W,M) \in THB$*, does there exist a solution to this problem with cost at most K?"*

3.3.2 THB-dec $\in NP$

The problem is in NP if there exists a polynomial-time verification procedure for all $(P = (B,C,F,G,H,W,M),K) \in THB-dec$. We will present this procedure below.

Given a problem (P,K) and a solution consisting of a set of trains T, and a set of trucks R that may participate in one of the batches in B, perform the following steps:

begin

- Calculate the costs according to the optimization cost function and verify these costs to be smaller than or equal to K	$O(log(C)*F)+O(log(K))$
- Verify that all containers are transported	$O(F)$
- Verify that no transfers occur between trains or trucks that are not part of the same batch	$O(F)$
- Verify that storage on the hub is consistent (i.e. there exist no transfers from any batch b_i to b_j with $i > j$)	$O(B*F)$

- Verify that the desired frequency is achieved $O(I*P)+O(F)$
- Verify that the schedule is periodical $O(I*P)+O(F)$
- Verify that the stack size never exceeds M $O(B*F)$

end

Total time complexity of the verification procedure: $O(log(C)*F)+O(log(K))+O(F)+O(F)+O(B*F)+O(I*P)+O(F)+O(I*P)+O(F)+O(B*F)=O(F*(log(C)+B)+O(log(K))+O(I*P))=O(|input|)$

This means that the problem is verifiable in polynomial time, so THB-dec $\in NP$

3.3.3 THB-dec $\in NPC$

We have shown that the problem is in NP. To prove that THB-dec is NP-Complete, we need to provide a polynomial reduction function transforming a problem instance of a problem that is known to be in NPC to a problem instance of THB-dec.

The NP-complete problem that will be used to show THB-dec is also NP-complete is the PARTITION-problem. This problem is one of the 21 problems proven NP-complete by Karp [22], and is defined as follows:

PARTITION:
"Given a set of items $A=a_0,a_1,\ldots,a_n$, and a weight function $g(a_i)\in\mathbb{N}$, is there a subset of items $I\subseteq A$ where $\sum_{a_i\in I} g(a_i)=\sum_{a_i\notin I} g(a_i)$?"

Before defining the reduction function that transforms an instance of the PARTITION problem to an instance of the THB-dec problem, we provide an intuitive description explaining the workings of this reduction function.

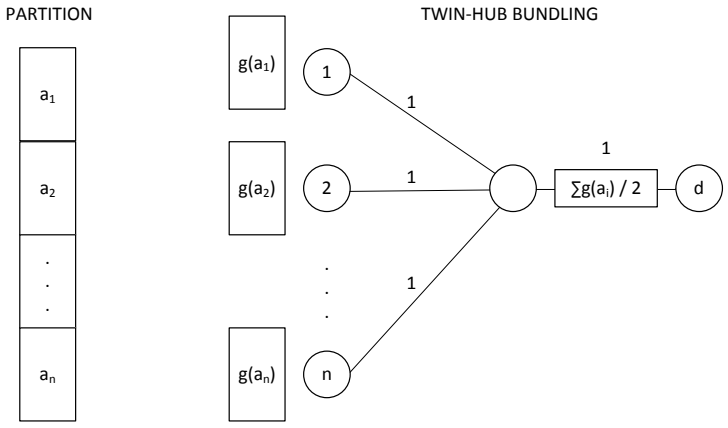

Figure 3.4: The concept behind the reduction function

Figure 3.4 shows the concept behind the reduction. A network is constructed with a hub and one inland terminal. Each entry a_i of the set A corresponds to a port terminal with a flow $g(a_i)$ from that terminal to the inland terminal. All distances to and from the hub are set to 1, the other costs are set to $|A|+3$, meaning that the trains must travel through the hubs if the costs are not to exceed $K = |A|+2$. The kilometre cost is then set to 1, and all other train costs are set to 0, making sure that a train travelling to or from the hub has a cost of 1. Trucks are effectively disabled by setting their costs higher than K.

Every train has a capacity of exactly half the sum of the weights of all elements in A, rounded down if the result of the division is a fraction. If the result of the division is a fraction, this means that the sum of the weights is odd and can therefore never be divided in two groups of equal size, meaning that we're dealing with a NO-instance of PARTITION. By rounding down, we make sure that there exists no schedule with cost $\leq K$, as at least three trains are needed to transport all containers from the hub to the destination.

Furthermore, at most $|A|-1$ trains may combine at the hub and exchange containers at the hub during one of the two batches, meaning that we cannot have a schedule in which all cargo is transported to the hub and divided over two trains that go to the inland terminal in one batch. The maximum stack size M is equal to zero, so no containers can be stored at the hub between the two batches.

Now that the concept behind the reduction function is clear, we formally define the reduction function f that reduces a problem in PARTITION to a problem in THB-dec:

input: (A,g), instance of PARTITION
output: $(P = (B,C,F,G,H,W,M),K)$ instance of THB-dec
begin

$I = \{d\}$
$P = \{v_i \mid \forall a_i \in A\}$
$V = I \cup P \cup \{h\}$
$\forall e \in E^{hub} : dist(e) = 1$
$\forall e \in E^{direct} : dist(e) = |A|+3$
$G = (V,E)$
$B = \{b_1, b_2\}$
$C_{km}^{fixed} = 1$
$C_{time}^{fixed} = C_{time}^{variable} = C_{km}^{variable} = C^{transfer} = C^{pph} = 0$
$C_{time}^{truck} = C_{km}^{truck} = |A|+3$
$F = \left\{F_{i,d} = g(a_i) \mid a_i \in A\right\}$
$H = |A|-1$
$W = \left\lfloor 0.5 * \sum_{a_i \in A} g(a_i) \right\rfloor$
$M = 0$

$K = |A| + 2$
return $(P = (B, C, F, G, H, W, M), K)$
end

Now that we have defined the reduction function f, we have to show that this reduction is correct and that it is verifiable in polynomial time. The three steps in proving the NP-completeness of THB-dec are therefore as follows.

1. $(A, g) \in$ PARTITION $\Rightarrow f(A, g) \in$ THB-dec
2. $f(A, g) \in$ THB-dec $\Rightarrow (A, g) \in$ PARTITION
3. f is computable in polynomial time

We will first show that $(A, g) \in$ PARTITION implies that $f(A, g) \in$ THB-dec, meaning that if we have a YES-instance of PARTITION – an instance where the items can be divided in two groups of equal weight – that the THB-dec instance created by the reduction function has a schedule in which $K \leq |A| + 2$.

As $(A, g) \in$ PARTITION, the set A can be divided in two disjoint subsets A_1 and A_2, with the weight of the items in A_1 equal to the weight of the items in A_2. When applying the reduction function on the PARTITION-instance, the created THB-dec instance has $|A|$ port terminals, each having a flow towards the inland terminal d of $g(a_i)$, with a_i the item corresponding to the terminal. Now create $|A_1|$ trains driving from the terminals corresponding to the elements in A_1 to the hub, transporting all containers available at these terminals. Note that, as $A_2 \notin \emptyset$, this requires at most $|A| - 1$ trains. The containers can therefore be transported to the hub, and the total amount of containers transported is exactly half of all containers. This means that we can pick one train from the trains driving towards the hub, load all arriving containers on that train, and let it continue its journey towards the inland terminal. Similarly, we create $|A_2|$ trains transporting all containers corresponding to elements in A_2 to the hub, and choose one to transport those containers to the inland terminal. This means that we have created $|A_1| + |A_2|$ trains that go to the hub with a cost of $|A_1| + |A_2| = |S|$, and have chosen two of them to continue to the inland terminal, incurring an additional cost of 2, making a total cost of $|A| + 2 \leq K$. This means that $f(A, g) \in$ THB-dec.

Now, we will show that $f(A, g) \in$ THB-dec implies that $(A, g) \in$ PARTITION, meaning that if we have a YES-instance of THB-dec that is produced by the reduction function – an instance where all flows can be transported with costs at most $K \leq |A| + 2$, that the instance (A, g) is a YES-instance of PARTITION.

As $f(A, g) \in$ THB-dec, there exist two batches $\{b_1, b_2\}$ with trains that transport all flows to the inland terminal d at a cost of at most $K = |A| + 2$. This solution contains at least $|A|$ trains, as for each port terminal there must

be a train that transports the flows to the hub, as the flow at each terminal is greater than zero. Note that these $|A|$ trains must be scheduled in two batches, as the maximum batch size is $|A|-1$. The costs incurred for transporting these containers towards the hub is $|A|$, meaning that we have $K-|A|=2$ 'left' for transporting all containers from the hub to d. This is enough to let at most 2 trains drive from the hub to d. As we're dealing with a YES-instance of THB-dec, these two trains are enough to transport all containers. But this means that we have two batches where each batch contains one train travelling from the hub to d transporting exactly half of the total amount of containers, as the capacity of the train is $W = \sum_{a_i \in A} g(a_i)/2$. Now create two sets of port terminals P_1 containing all port terminals with a train transporting the containers of that terminal in batch b_1 and P_2 containing those terminals serviced in b_2. We choose the items corresponding to the terminals in P_1 to be included in set A_1, and those corresponding to P_2 to be included in A_2. We now have two disjoint sets of items, with $A_1 \cup A_2 = A$, meaning that $(A,g) \in$ PARTITION.

It is obvious that the reduction function f can be computed in polynomial time. This means that we have proven that THB-dec $\in NPC$.

3.3.4 Problems with stack sizes greater than zero

In Section 3.3.3, a reduction function was presented that disabled the stack. However, the proof still holds for any finite stack size M. The following modification to the reduction function needs to be made for a given M:

- $F = \left\{ F_{i,d} = g(a_i) * (M+1) \mid a_i \in A \right\}$
- $W = \lfloor \sum_{a_i \in A} g(a_i)/2 \rfloor * (M+1)$

Even though containers can be placed on the stack now, only M containers can be placed on the stack, which is one container less than the equivalent of one 'unit' in the PARTITION-problem. As only two batches (b_1 and b_2) are allowed through the reduction function f, the maximum amount of containers transferred through the stack from batch b_1 to batch b_2 would be M. Through the modification presented above, this leaves at least 1 container at each terminal from which a container is put on the stack. In order to transport all containers, a second train is then required from the same terminal(s) in batch b_2. This means that at least $|A|+1$ trains are required to transport all containers towards the hub, as there exists a source terminal for which a train is scheduled in batch b_1 as well as in batch b_2. As at least 2 trains are required to transport all containers from the hub to the destination terminal, the total costs would be at least $|A|+3 > K$, making it impossible to transfer the equivalent of one container in the original THB-dec problem through the stack. Therefore, the stack size has no effect on the reduction when the above modification is made to accommodate for the stack size.

3.4 Discussion

In this chapter, we have analysed the Twin Hub Bundling (THB) problem, and have analysed the algorithmic complexity of this problem. We have proven that the Twin Hub Bundling (THB) problem is an NP-hard optimisation problem by showing that the decision variant of this problem (THB-dec) is NP-Complete. This means that any algorithm solving this problem to optimality will presumably take an exponential amount of time. In other words, it is very unlikely that an efficient algorithm can be designed that solves the THB problem to optimality in a reasonable amount of time. Finding the optimal solution to smaller problem instances may still be feasible, but solving larger instances to optimality may take centuries; even on the fastest computers available today.

As we do not have the luxury of waiting centuries for an algorithm to find the optimal bundling networks for a given problem instance, we must aim to develop an algorithm that finds good-quality solutions within a reasonable amount of time. In the next chapter, we will investigate which kinds of algorithms may be used, and present an algorithm to solve the THB problem.

Chapter 4

Algorithms

In the previous chapter, we have seen that the Twin Hub Bundling (THB) problem is an NP-hard problem. This means that it is unlikely that an efficient algorithm can be designed that solves the THB problem to optimality within a reasonable amount of time. The performance requirements for the bundling tool state that the algorithm may take at most 24 hours to complete (see Section 1.3.1), which means that we must focus on finding a solution that is as close to optimality as possible.

Given enough time, exact algorithms can solve the problem to optimality, though it is crucial that these algorithms are anytime algorithms, i.e. they can be stopped at any time during the computation process (e.g. 24 hours) to return the best solution found so far. An Integer Linear Programming (ILP) approach can be used to create such an algorithm. It has the additional property that it can produce a lower bound on the solution quality, which can be used as a quality measure for the solutions it produces. The ILP-algorithm is, therefore, a good candidate for solving the THB-problem and meeting the performance requirements.

In addition to the ILP-algorithm, we will also introduce a relaxation of the ILP-formulation that can be used to find a better initial solution for the (exact) ILP-algorithm. This relaxation will exclude certain parts of the solution space, meaning that the solution to the relaxation is not the optimal solution to the THB-problem; essentially trading solution quality for speed. This solution can then be used as a start solution for the exact algorithm, which might cause a speed-up and allow the algorithm to find a better solution after 24 hours.

In this chapter, we will present both the ILP-algorithm and the relaxation of the problem. Computational results of both the ILP-algorithm and the combination of the relaxation and the ILP-algorithm can be found in Section 5.

4.1 Integer Linear Programming algorithm

Integer Linear Programming (ILP) is a method for solving optimization problems that yields an exact solution if given enough processing time. It requires

the optimization problem to be formulated using a mathematical model containing a linear objective function over a set of decision variables, and a set of linear constraints on those decision variables specifying the conditions the solution must adhere to.

The objective function and the constraints specify the solution space, in which the optimal solution must be found. This solution consists of the assignment to the decision variables that yields the lowest objective value (or highest, depending on the optimization direction). The challenge of developing an ILP-algorithm for a problem lies therefore in the transformation of the THB-problem to an ILP-problem. In other words, we must create an objective function and constraints over decision variables, such that the solution to this problem represents a set of optimal Twin-Hub Bundling networks.

4.1.1 Modelling the THB-problem as an ILP-problem

The definition of the Twin-Hub Bundling (THB) problem (see Section 3.2) specifies the problem to be solved with the Integer Linear Programming algorithm. The THB-problem specifies a minimization problem, where train and truck services need to be created that transport all container flows in the cheapest way possible. Each aspect of this THB-problem must be modelled as an Integer Linear Programming problem such that the solution found can be transformed into a solution for the THB-problem.

Modelling transport service options

The transport service options (i.e. the trains and trucks) form the core part of the desired THB-solution – the goal is to find Twin-Hub networks consisting of a combination of train and truck services that meet at the same time. Each of these transport service options must have a representation in the ILP-formulation. However, there is no ready-made concept of a transportation network available in Integer Linear Programming. This means that each transport option must be enumerated in order to obtain a representation of that option.

Each transport option represents a part of the journey of a train or truck in the network. For example, direct trains provide a means of transporting containers from one source to one destination, while hub trains only take the containers for the part from the source terminal to the hub. The costs for letting a train drive with a certain number of containers can be known beforehand, as the costs for this train are divided into a fixed part per train (fixed hour and kilometre costs) and a variable cost per container that is transported by that train (variable hour and kilometre costs, pre- and/or post-haulage costs and transfer costs at the hub). Whether the pre- and/or post-haulage costs are incurred by a container travelling on a particular train is also known for each train, as we define each train in the ILP-model to have a fixed route. All possible train routes are enumerated in the ILP-model, so it can be decided whether or not to

run this particular service by examining the value of the corresponding decision variable in the solution.

Another issue that needs to be modelled in the ILP-formulation is the order of trains and trucks that visit the hub. For direct trains and trucks, order is not important, as these do not interact with each other. For trains that visit the hub, it is important to model the order in which the trains arrive, as there are dependency relations between these trains: one train may exchange containers with the stack or with another train present at the hub. Each train must therefore be assigned a batch number, which denotes the order in which the trains are serviced at the hub. As each transportation option visiting the hub may be assigned any batch number, all transportation options must be enumerated for all possible batch numbers in the ILP-model. This means that the number of batches allowed in this problem should be chosen conservatively, as an additional batch option increases the model size significantly.

Modelling containers

Now that we have defined all transportation options for all batches, we continue with the modelling of the containers. Each container must be assigned to a (combination of) transportation option(s), which transport that container from its source to its destination. As the variable costs are dependent on the distance and time travelled of that container on a certain train or truck, we can calculate these costs based on how many containers are transported using each transport option. There is, of course, a maximum number of containers that can be transported using each transport option, and this is linked to the number of trains or trucks that drive a service. A constraint should therefore be created that limits the number of containers transported by a certain option to the joint capacity offered on that service. Another constraint is needed to ensure that the total number of containers transported from a source terminal to a destination terminal is equal to the amount specified in the flow matrix, so the resulting solution is valid.

Modelling hub operations

The last step involves the modelling of the operations taking place at the hub. Constraints must be placed on the number of trains that may participate in each batch, which is dictated by the number of tracks at the hub that have overhead cranes to transfer containers from one train to the other, and on the total number of containers that can be placed on the stack. Accounting for the transfers taking place at the hub is easy for the hub and destination trains and trucks, as they respectively unload and load all containers at the hub. Each container arriving at the hub incurs half of the total transfer costs, and the other half is incurred when the container is loaded on the train or truck to the destination. For indirect trains, the modelling is more difficult, as not all containers on the train are transferred. Distinguishing these containers is important, as transfer

costs should only be incurred by those containers that are actually transferred. This is modelled by distinguishing between the destination of the container and the destination of the train itself. If these are different, then the container is apparently transferred to another train or to the stack. This does mean, however, that we must create additional variables for each indirect train: one for each destination that a container on that train can travel to.

Modelling constraints

Now that we have created all required constraints and include all costs in the objective function, we can use this formulation to compute bundling networks. To ensure the periodicity of the generated schedule, one full period is generated, both starting from an empty stack at the hub and ending with an empty stack. This also limits the size of the model, as using multiple time periods would increase the size of the model beyond the point at which it is realistically solvable. To reduce the size of the model even further, we use the property of the problem that containers from different distance classes can never combine in one batch, as these will never be able to conform to the periodicity constraint. This means that we can treat each distance class (i.e. one-day, two-day, and multi-day round-trips) as a subproblem that can be solved independently of the other subproblems. The result of each subproblem can then be combined to form the complete network.

4.1.2 Integer Linear Programming formulation

From the modelling steps in Section 4.1.1 we can create a detailed ILP-formulation. This formulation closely follows the THB problem definition as presented in Section 3.2 and the full formulation presented in Appendix A, and consists of three parts: a table of input parameters (Table 4.1), a table of decision variables (Table 4.2), and a table containing the objective function and the constraints (Table 4.3). Each of these parts will be presented and discussed below.

Input parameters

The input parameters (Table 4.1) specify the network structure of the THB-problem for the ILP-algorithm, as well as all costs and capacity parameters. The network structure is specified by creating two sets of variables: the set of origins I and destinations J. This formulation is independent of the THB-formulation in the sense that I (and J) may either consist of the inland terminals or the port terminals in the THB-formulation.

The flows between these sets of origin and destination terminals is specified by the matrix F. The distances between the terminals do not need to be specified, as the distances only matter for the calculation of the costs between terminals. These costs are specified for each transport option, for each origin/destination pair, and consist of the fixed cost per train (f), variable costs

Parameter		**Description**
I	=	set of origins
J	=	set of destinations
B	=	list of batches
f_{ij}^{direct}	=	fixed cost of running a train directly between $i \in I$ and $j \in J$
$f_{ij}^{indirect}$	=	fixed cost of running a train between $i \in I$ and $j \in J$ while visiting the hub
f_i^{hub}	=	fixed cost of running a train between $i \in I$ and the hub
f_j^{dest}	=	fixed cost of running a train between the hub and $j \in J$
v_{ij}^{direct}	=	variable unit cost of transporting a container directly from $i \in I$ to $j \in J$, including pre- and post-haulage
$v_{ij}^{indirect}$	=	variable unit cost of transporting a container from $i \in I$ to $j \in J$ while visiting the hub, including pre- and post-haulage
v_i^{hub}	=	variable unit cost of transporting a container from $i \in I$ to the hub, including pre-haulage
v_j^{dest}	=	variable unit cost of transport a container from the hub to $j \in J$, including post-haulage
R_{ij}^{direct}	=	unit cost of transporting a container directly from $i \in I$ to $j \in J$ by truck
R_i^{hub}	=	unit cost of transporting a container from $i \in I$ to the hub by truck
R_j^{dest}	=	unit cost of transporting a container from the hub to $j \in J$ by truck
$C^{transfer}$	=	cost of transferring a container from one transport to another at the hub
W	=	maximum capacity of a train
H	=	maximum number of trains that can attend the hub simultaneously in a batch
M	=	maximum number of containers that can be stored on the stack
F_{ij}	=	number of containers to be transported from $i \in I$ to $j \in J$

Table 4.1: Parameters of the ILP problem

per container on a train (v) and the cost of transporting a container per truck (r). As described in the modelling, these costs can be calculated for each transport option beforehand. Therefore, the costs for each train and truck on a specific route needs to be calculated only once, based on the cost model (hour and kilometre coefficients) as specified in the THB-problem.

The transfer costs at the hub ($C^{transfer}$), the list of possible batches B, as well as the number of trains that may partake in a batch H can be copied directly from the THB-formulation.

Decision variables

The decision variables (Table 4.2) specify the trains and trucks that can be generated, and the number of containers that are stored at the hub and transferred

Decision variable		Description	Bounds
t_{ij}^{direct}	=	number of trains driving directly between $i \in I$ and $j \in J$	$[0, \lceil F_{ij}/W \rceil]$
$t_{ijb}^{indirect}$	=	number of trains driving between $i \in I$ and $j \in J$ while visiting the hub belonging to batch $b \in B$	$[0,1]$
t_{ib}^{hub}	=	number of trains driving between $i \in I$ and the hub belonging to batch $b \in B$	$[0, \min(\sum_{j \in J} \lceil F_{ij}/W \rceil, H)]$
t_{jb}^{dest}	=	number of trains driving between the hub and $j \in J$ belonging to batch $b \in B$	$[0, \min(\sum_{i \in I} \lceil F_{ij}/W \rceil, H)]$
c_{ij}^{direct}	=	number of containers transported from $i \in I$ to $j \in J$ by a direct train	$[0, F_{ij}]$
$c_{ijb}^{indirect}$	=	number of containers transported from $i \in I$ to $j \in J$ by an indirect train visiting the hub in batch $b \in B$ where no transfer is required	$[0, \min(F_{ij}, W)]$
$c_{ijj'b}^{indirect\_transfer}$	=	number of containers transported from $i \in I$ to $j' \in J$, being transported on an indirect train headed for destination $j \in J$, and visiting the hub in $b \in B$	$[0, \min(F_{ij}, W)]$
$c_{ijb}^{indirect\_hub}$	=	number of containers transported from $i \in I$ to the hub destined for $j \in J$, belonging to batch $b \in B$, on an indirect train	$[0, \min(F_{ij}, W)]$
$c_{ijb}^{indirect\_dest}$	=	number of containers transported from the hub to $j \in J$, having origin $i \in I$ and belonging to batch $b \in B$, on an indirect train	$[0, \min(F_{ij}, W)]$
c_{ijb}^{hub}	=	number of containers transported from $i \in I$ to the hub destined for $j \in J$, belonging to batch $b \in B$, on a train ending at the hub	$[0, \min(F_{ij}, W)]$
c_{ijb}^{dest}	=	number of containers transported from the hub to $j \in J$, having origin $i \in I$ and belonging to batch $b \in B$, on a train starting at the hub	$[0, \min(F_{ij}, W)]$
r_{ij}^{direct}	=	number of containers transported directly from $i \in I$ to $j \in J$ by truck	$[0, F_{ij}]$
r_{ijb}^{hub}	=	number of containers transported from $i \in I$ to the hub by truck, destined for $j \in J$, belonging to batch $b \in B$	$[0, F_{ij}]$
r_{ijb}^{dest}	=	number of containers transported from the hub to $j \in J$ by truck, having origin $i \in I$ and belonging to batch $b \in B$	$[0, F_{ij}]$
$c_b^{transfer}$	=	number of containers transferred from one transport (train or truck) to another at the hub during batch $b \in B$	$[0, H \times W]$
A_{ijb}^{hub}	=	number of containers that are stored after batch $b \in B$ at the hub with origin $i \in I$ and destination $j \in J$	$[0, \min(W, F_{ij})]$

Table 4.2: Decision variables for the ILP problem

at the hub. As mentioned in the modelling, each transport option for each batch is generated and specified in the formulation. The variables t enumerate all the trains, and the variables r enumerate all the trucks. The number of containers transported on each train segment are specified by c, with the indirect trains requiring an extensive modelling (4 types of variables) to distinguish between containers that stay on a train ($c^{indirect}$), containers that are transferred onto another indirect train ($c^{indirect\_transfer}$), containers that stay on an indirect train until the hub ($c^{indirect\_hub}$), and containers that are transferred on an indirect train to its destination ($c^{indirect\_dest}$). For each batch, the amount of transferred containers is calculated in $c^{transfer}$, and the amount of containers stored on the stack is kept in A.

For each decision variable, a range of allowed values has to be specified. The lower bound for all decision variables is zero, as each decision variable represents running a transport service, or transporting a container, all of which can only take non-negative values. The upper bounds on each variable should be as tight as possible to limit the size of the solution space and speed up the solution process. Proofs for non-obvious upper bound values stated in Table 4.2 can be found in Appendix B.

Objective function and constraints

The objective function and the constraints (Table 4.3) form the core of the ILP-formulation. The objective function describes the function to be optimized. In this case, the sum of all costs involved in transporting the containers through the network needs to be minimized. This means that for each transport option, for each batch, the costs for that option need to be included in the objective function. The costs for transporting each container on a train, which includes the variable train costs and the pre- and post-haulage costs for that container, are also included in the objective function. Lastly, the cost of the hub transfers are also included in the objective function.

Without constraints on the decision variables, the specified objective function would always yield zero, as no trains or trucks would be generated. The constraints below force the solution to the ILP-model to represent a valid solution to the THB-problem. Each constraint in the formulation will be explained below:

The first constraint (4.1.1) specifies that all containers in the system must be transported on some form of transport. This is specified for each source/destination pair, so each flow is accounted for. A container transported to the hub accounts for half of the total amount transported, so that when paired with a transport to the destination – which also counts for half a transported container – this counts as one transported container. Constraint 4.1.2 specifies that each container transported to the hub must be matched by a transport to the correct destination in other (later) batches.

Constraints 4.1.3, 4.1.4, and 4.1.5 ensure that the hub inventory of contain-

ers is accounted for correctly. The first constraint initializes the hub inventory for all routes to 0, as we assume there is no inventory before the first batch. The second and third constraint ensure that the inventory is consistent across the batches. Containers can only be exchanged between batches if the batch number at which the container is delivered at the hub is smaller than or equal to the batch number in which the container is transported to its destination.

We must ensure the train capacity is sufficient to transport all containers through the network. Constraints 4.1.6, 4.1.7, and 4.1.8 ensure that if a container is designated to be transported on a direct train, a hub train, or a destination train respectively, there is enough capacity reserved on a train to transport that container.

For the containers designated to be transported on an indirect train, determining how many trains must drive is less straight-forward. There must be enough capacity available in a batch to transport the containers from the origins to the hub (Constraint 4.1.10), and to transport the containers from the hub to the destinations (Constraint 4.1.11). These constraints would be sufficient to guarantee capacity for all containers if transfers would not need to be modelled. To model these transfer costs, we create two categories of containers: one which incurs transfer costs and one which does not. We then restrict the group which does not incur transfer costs to contain only those containers of which the origin and destination match the origin and destination of the train, and calculate transfer costs for those containers that do not. Constraint 4.1.9 is therefore needed to ensure that this happens.

Constraint 4.1.12 ensures that in each batch not more than H trains may partake, as there are only a limited number of tracks available at the hub. There is no such limit on trucks, however, as trucks can also drop off and pick up their cargo in between batches.

Constraint 4.1.13 is needed to account for the transfer costs. The transfer costs are incurred by all containers that are transported by trucks and trucks originating from or ending at the hub, as well as for all containers that are transferred to and from indirect trains, i.e. those containers in “indirect transfer”.

The final constraint (4.2.11) specifies that the stack size must never exceed the stack capacity M.

ILP formulation:

$$\begin{aligned}\text{minimize } Z = & \sum_{i\in I, j\in J} f_{ij}^{direct} t_{ij}^{direct} + \sum_{i\in I, j\in J, b\in B} f_{ij}^{indirect} t_{ijb}^{indirect} + \sum_{ib} f_i^{hub} t_{ib}^{hub} + \sum_{jb} f_j^{dest} t_{jb}^{dest} \\ & + \sum_{i\in I, j\in J} v_{ij}^{direct} c_{ij}^{direct} + \sum_{i\in I, j\in J, b\in B} v_{ij}^{indirect} (c_{ijb}^{indirect} + \sum_{j'\in J, j'\neq j} c_{ijj'b}^{indirect\_transfer}) \\ & + \sum_{i\in I, j\in J, b\in B} v_i^{hub} (c_{ijb}^{hub} + c_{ijb}^{indirect\_hub}) + \sum_{i\in I, j\in J, b\in B} v_j^{dest} (c_{ijb}^{dest} + c_{ijb}^{indirect\_dest}) \\ & + \sum_{i\in I, j\in J} R_{ij}^{direct} r_{ij}^{direct} + \sum_{i\in I, j\in J, b\in B} R_i^{hub} r_{ijb}^{hub} + \sum_{i\in I, j\in J, b\in B} R_j^{dest} r_{ijb}^{dest} + \sum_b C^{transfer} c_b^{transfer}\end{aligned}$$

Subject to:

$$\begin{aligned}\sum_{b\in B} (c_{ijb}^{indirect} + \sum_{j'\in J, j'\neq j} c_{ij'jb}^{indirect\_transfer} + 0.5c_{ijb}^{hub} + 0.5c_{ijb}^{dest} + 0.5r_{ijb}^{hub} + 0.5r_{ijb}^{dest} \\ + 0.5c_{ijb}^{indirect\_hub} + 0.5c_{ijb}^{indirect\_dest}) + c_{ij}^{direct} + r_{ij}^{direct} = F_{ij} \quad \forall (i\in I, j\in J)\end{aligned} \tag{4.1.1}$$

$$\sum_{b\in B} (c_{ijb}^{hub} + r_{ijb}^{hub} + c_{ijb}^{indirect\_hub} - c_{ijb}^{dest} - r_{ijb}^{dest} - c_{ijb}^{indirect\_dest}) = 0 \quad \forall (i\in I, j\in J) \tag{4.1.2}$$

$$\sum_{i\in I, j\in J} A_{ijb}^{hub} = 0 \quad b = B_0 \tag{4.1.3}$$

$$c_{ijb}^{hub} + r_{ijb}^{hub} + c_{ijb}^{indirect\_hub} - A_{ijb}^{hub} - c_{ijb}^{dest} - r_{ijb}^{dest} - c_{ijb}^{indirect\_dest} = 0 \quad \forall (i\in I, j\in J), b = B_0 \tag{4.1.4}$$

$$\begin{aligned}A_{ij(b-1)}^{hub} + c_{ijb}^{hub} + r_{ijb}^{hub} + c_{ijb}^{indirect\_hub} - A_{ijb}^{hub} - c_{ijb}^{dest} \\ - r_{ijb}^{dest} - c_{ijb}^{indirect\_dest} = 0 \quad \forall (i\in I, j\in J, b\in B | b\neq B_0)\end{aligned} \tag{4.1.5}$$

$$c_{ij}^{direct} \leq W t_{ij}^{direct} \quad \forall (i\in I, j\in J) \tag{4.1.6}$$

$$\sum_{j\in J} c_{ijb}^{hub} \leq W t_{ib}^{hub} \quad \forall (i\in I, b\in B) \tag{4.1.7}$$

$$\sum_{i\in I} c_{ijb}^{dest} \leq W t_{jb}^{dest} \quad \forall (j\in J, b\in B) \tag{4.1.8}$$

$$c_{ijb}^{indirect} + \sum_{j'\in J, j'\neq j} c_{ijj'b}^{indirect\_transfer} \leq W t_{ijb}^{indirect} \quad \forall (i\in I, j\in J, b\in B) \tag{4.1.9}$$

$$\sum_{j'\in J} (c_{ij'b}^{indirect\_hub} + \sum_{j\in J, j\neq j'} c_{ijj'b}^{indirect\_transfer}) + \sum_{j\in J} c_{ijb}^{indirect} \leq \sum_{j\in J} W t_{ijb}^{indirect} \quad \forall (i\in I, b\in B) \tag{4.1.10}$$

$$\sum_{i\in I} (c_{ijb}^{indirect} + \sum_{j'\in J, j'\neq j} c_{ij'jb}^{indirect\_transfer} + c_{ijb}^{indirect\_dest}) \leq \sum_{i\in I} W t_{ijb}^{indirect} \quad \forall (j\in J, b\in B) \tag{4.1.11}$$

$$\sum_{i\in I, j\in J} t_{ijb}^{indirect} + \sum_i t_{ib}^{hub} + \sum_j t_{jb}^{dest} \leq H \quad \forall b\in B \tag{4.1.12}$$

$$\begin{aligned}\sum_{i\in I, j\in J} 0.5c_{ijb}^{hub} + 0.5c_{ijb}^{indirect\_hub} + 0.5c_{ijb}^{dest} + 0.5c_{ijb}^{indirect\_dest} + 0.5r_{ijb}^{hub} + 0.5r_{ijb}^{dest} \\ + \sum_{j'\in J, j'\neq j} c_{ijj'b}^{indirect\_transfer} = x_b^{transfer} \quad \forall b\in B\end{aligned} \tag{4.1.13}$$

$$\sum_{i\in I, j\in J} A_{ijb}^{hub} \leq M \quad \forall b\in B \tag{4.1.14}$$

Table 4.3: Integer Linear Programming formulation of the problem. The bounds for the decision variables are given in Table 4.2.

4.1.3 Solving the ILP-model of a THB-problem

The ILP-formulation specifies the solution space to be searched using the constraints and the variable bounds. The objective function provides a measure for the 'fitness' of each valid solution in the solution space. When solving the ILP-model for a certain THB-problem instance, the solver should find the solution in the solution space that has the smallest objective value, and so the smallest costs of the THB-problem solution. Finding the solution to 0-1 Integer Programming – a decision problem – was proven to be an NP-Complete problem by Karp [22], which means that the optimisation variant is NP-hard. Any solver for this problem has therefore a worst-case exponential time complexity, unless $P = NP$.

The most well-known method for solving Integer Linear Programming problems is the Branch-and-Bound method. This method uses a Linear Programming solver to solve the problem without the integer constraints. Removing these constraints yields a problem that can be solved in polynomial time using interior-point methods, such as the projective method developed by Karmarkar [21], or in (worst-case) exponential time using the simplex method [11], which is usually faster on smaller problems. By solving the problem without integer constraints, a lower bound on the solution quality is found, as the continuous solution may contain solutions that are 'cut off' when integer constraints are added. The Branch-and-Bound algorithm then 'branches' on that solution by picking one integer variable that is not integer in the continuous solution, and creates two subproblems. In one subproblem, the variable is bounded to be greater than the integer value closest to the continuous value (rounded up). In the other subproblem, the variable is bounded to be smaller than integer value closest to the continuous value (rounded down). Both subproblems are then solved using the continuous method, yielding another lower bound on each of the subproblems. If that lower bound is larger than a known integer solution that has already been found, there is no point in further investigating that part of the solution space. That branch is then 'cut off'. Once all branches cannot yield a better solution than the integer solution that has been found up till then, the search can be stopped and that integer solution can be returned as the optimum.

There are several other methods for solving Integer Linear Programming, such as cutting plane-methods or branch-price-and-cut methods, and many heuristics have been developed for speeding up the solution process. Creating better Integer Linear Programming solvers has become a competitive industry, with CPLEX [19] and Gurobi [16] being the state-of-the-art solvers at the moment. Using one of these solvers is therefore the preferred option for solving the generated ILP-models. More details about the solver that is used can be found in Section 5.1. The computational results for solving THB-problems using this solver can be found in Chapter 5.

4.2 Relaxation of the ILP-formulation

The Integer Linear Programming (ILP) algorithm (Section 4.1) can find an exact solution for the Twin-Hub Bundling (THB) problem, but takes a very long time to complete, as we can see in the experimental evaluation of this algorithm in Section 5.1.3. If an initial solution with a better objective value could be found, this solution then will allow the ILP-solver to prune parts of the search space early on in the solution process, leaving more time to explore other, more promising parts in order to find a better-quality solution.

One of the ways to find an initial solution faster is to reduce the size of the ILP formulation by removing some of the constraints and variables from the model without invalidating the solution, i.e. creating a relaxation of the original ILP-formulation of the THB-problem. The relaxation must still yield a valid solution, meaning that constraints and variables required to keep the solution valid – such as constraints managing the hub inventory or the amount of containers that need to be transported – cannot be left out. Removing certain transport options from the range of possibilities is, therefore, the only way to reduce the model size. This obviously impacts the solution quality, as the options left out of the model cannot be chosen any more. As the ILP solver uses an algorithm that is exponential in the problem size, creating a relaxation can yield solutions much faster, essentially trading in solution quality for speed.

Choosing which transport option to leave out of the formulation is, however, not straightforward: any option that is left out causes a certain speedup, but also impacts the solution quality. In the discussion of the modelling of the THB-problem as an ILP-problem (Section 4.1.1), the indirect trains were identified as the hardest part to model, requiring four types of variables to model the different options for a container to travel on an indirect train. Analysis of the model shows that modelling one of these options – the containers that are transferred between two indirect trains $c^{indirect\_transfer}$ – requires a number of variables quadratic in the number of destination terminals. Leaving out the indirect trains as an option reduces the size of the ILP-model for real-world sized problems by more than half. The containers otherwise transported using indirect trains can still benefit from hub-and-spoke bundling by taking hub- and destination trains and trucks instead. Making an ILP-formulation without indirect trains would, therefore, make a valid relaxation.

Table 4.4 shows the resulting ILP formulation of the relaxation, where the parts of the original formulation concerning the indirect trains have been removed. Removing the indirect trains from the model has led to a much more compact formulation, in which the transfers of containers at the hub have been greatly simplified, as all containers now incur transfer costs.

The downside of removing indirect trains is that the optimal solution of the relaxation has a higher cost than the optimal solution of the THB-problem, and that this solution is not satisfactory for solving the THB-problem, as no bundling networks are identified. Bundling networks consist of indirect trains by

ILP formulation:

$$\text{minimize } Z = \sum_{i\in I, j\in J} f_{ij}^{direct} t_{ij}^{direct} + \sum_{ib} f_i^{hub} t_{ib}^{hub} + \sum_{jb} f_j^{dest} t_{jb}^{dest} + \sum_{i\in I, j\in J} v_{ij}^{direct} c_{ij}^{direct}$$
$$+ \sum_{i\in I, j\in J, b\in B} c_i^{hub} c_{ijb}^{hub} + \sum_{i\in I, j\in J, b\in B} c_j^{dest} c_{ijb}^{dest} + \sum_{i\in I, j\in J} R_{ij}^{direct} r_{ij}^{direct}$$
$$+ \sum_{i\in I, j\in J, b\in B} R_i^{hub} r_{ijb}^{hub} + \sum_{i\in I, j\in J, b\in B} R_j^{dest} r_{ijb}^{dest} + \sum_b C^{transfer} c_b^{transfer}$$

Subject to:

$$0.5c_{ijb}^{hub} + 0.5c_{ijb}^{dest} + 0.5r_{ijb}^{hub} + 0.5r_{ijb}^{dest} + c_{ij}^{direct} + r_{ij}^{direct} = F_{ij} \quad \forall (i \in I, j \in J) \tag{4.2.1}$$

$$\sum_{b\in B} (c_{ijb}^{hub} + r_{ijb}^{hub} - c_{ijb}^{dest} - r_{ijb}^{dest}) = 0 \quad \forall (i \in I, j \in J) \tag{4.2.2}$$

$$\sum_{i\in I, j\in J} A_{ijb}^{hub} = 0 \quad b = B_0 \tag{4.2.3}$$

$$c_{ijb}^{hub} + r_{ijb}^{hub} - A_{ijb}^{hub} - c_{ijb}^{dest} - r_{ijb}^{dest} = 0 \quad \forall (i \in I, j \in J), b = B_0 \tag{4.2.4}$$

$$A_{ij(b-1)}^{hub} + c_{ijb}^{hub} + r_{ijb}^{hub} - A_{ijb}^{hub} - c_{ijb}^{dest} - r_{ijb}^{dest} = 0 \quad \forall (i \in I, j \in J, b \in B | b \neq B_0) \tag{4.2.5}$$

$$c_{ij}^{direct} \leq W t_{ij}^{direct} \quad \forall (i \in I, j \in J) \tag{4.2.6}$$

$$\sum_{j\in J} c_{ijb}^{hub} \leq W t_{ib}^{hub} \quad \forall (i \in I, b \in B) \tag{4.2.7}$$

$$\sum_{i\in I} c_{ijb}^{dest} \leq W t_{jb}^{dest} \quad \forall (j \in J, b \in B) \tag{4.2.8}$$

$$\sum_i t_{ib}^{hub} + \sum_j t_{jb}^{dest} \leq H \quad \forall b \in B \tag{4.2.9}$$

$$\sum_{i\in I, j\in J} (0.5c_{ijb}^{hub} + 0.5c_{ijb}^{dest} + 0.5r_{ijb}^{hub} + 0.5r_{ijb}^{dest}) = x_b^{transfer} \quad \forall b \in B \tag{4.2.10}$$

$$\sum_{i\in I, j\in J} A_{ijb}^{hub} \leq M \quad \forall b \in B \tag{4.2.11}$$

Table 4.4: Relaxation of the Integer Linear Programming formulation. The bounds for the decision variables are given in Table 4.2.

definition, and leaving out this option means that any solution to the relaxation does not meet the requirements of the bundling tool. This is why the relaxation can only be used in combination with the ILP-algorithm.

This is not a problem, however, as it is expected that the solution to the relaxation can be found much faster. In the first minutes of computation time, the algorithm using the relaxation may be able to find a solution with much lower costs than the ILP-algorithm can. This effect is expected to be most pronounced for larger models where the number of destinations is high. The performance of the combination of the algorithm using both the relaxation and

the full ILP-formulation is investigated in Chapter 5.

4.3 Discussion

Two algorithms have been presented in this chapter; one based on Integer Linear Programming (ILP) capable of solving the THB-problem, and meeting all the requirements for a valid solution, and a relaxation of the ILP-formulation, which does not contain the variables and constraints associated with indirect trains. The solutions found using the relaxation are, therefore, not valid solutions to the THB-problem as these indirect trains are crucial to the THB problem. However, the solutions found using the relaxation are valid starting solutions for the full ILP algorithm, and may yield a speed-up of the full ILP algorithm by allowing the solver to prune certain part of the search space faster. The performance will therefore be analysed experimentally in the next chapter.

Chapter 5

Computational analysis

Two algorithms were specified in Chapter 4 for solving the Twin Hub Bundling (THB) problem: an exact anytime algorithm based on Integer Linear Programming (ILP) and a relaxation of the ILP-formulation that can be used to provide an initial solution for the exact algorithm, and possibly cause a speed-up. This combination will be called the *cascade algorithm*.

It is still unclear whether either of the two algorithms meet the performance requirements of the bundling tool (Section 1.3.1), especially since both algorithms may take exponential time to finish. Therefore, we will investigate the performance of the algorithms experimentally, starting by answering the following questions for the ILP-algorithm:

1. **What is the performance on the real-world instance?** Evaluation of the performance on the real-world instance is required to decide whether the algorithm meets the performance requirements of the bundling tool.
2. **What instances can be solved to optimality within a reasonable time frame?** The target is to solve the problem instance as close to optimality as possible. Finding the limit at which instances can be solved to optimality within a reasonable amount of time can give not only give an indication of the performance that we should expect, but also allows for investigating which of the three factors (number of sources, number of destinations, number of batches) determine the performance of the algorithm most.

After the performance data has been gathered for the ILP-algorithm, we can perform the same tests on the cascade algorithm to investigate whether it performs better than the ILP-algorithm. This is done by answering the following questions:

3. **Does the cascade algorithm solve larger instances to optimality than the ILP-algorithm in the same time?**
4. **Does the cascade algorithm perform better on the real-world instance than the ILP-algorithm?**

Finally, we will perform two tests about certain properties of the problem instance that might influence the time to reach a certain solution quality:

5. **Does the asymmetry in the formulation of indirect trains have a significant influence on the runtime?** There exists an asymmetry in the formulation, as accounting for the transfers between indirect trains requires a number of variables linear in the number of sources and quadratic in the number of destinations. Investigating whether this asymmetry has a significant influence may lead to the development of an optimisation, where instances with more destinations than sources can be mirrored to reduce the size of the model, possibly leading to a speed-up.
6. **What is the influence of the choice of a maximum number of batches on the performance?** The choice of the maximum number of batches has to be made before starting the algorithm, even though it might be hard to estimate the number of batches in the optimal solution. Investigating what the effect of running the algorithm with a higher maximum number of batches is can lead to more insight into what maximum should be chosen.

Answering these questions will lead to a better understanding of the performance of both algorithms, and possibly lead to new ways of improving the performance of the algorithms.

In order to find these answers, we will first describe the test set-up used for all tests, after which each question is answered. Finally, a discussion about the results is presented.

5.1 Test set-up

All tests were run on the same machine using the same problem instances in order to make a realistic comparison between algorithms. In this section, the hard- and software and the problem instances used are described in more detail.

5.1.1 Hard- and software

Unless otherwise specified, the algorithms were run on a machine with a dual quadcore 2.33GHz Intel Xeon CPU (E5345 * 2) and 16 GB of memory running Fedora 15 64-bit. All processing cores were available to the algorithms, although memory usage was limited to 12GB to allow enough memory for system processes and prevent swapping to disk from interfering with the runtime results. The installed Java version was OpenJDK 7.

5.1.2 Problem instances

One real-world instance was provided by the OTB Research Group to test the algorithm with. This instance has 18 source terminals and 50 destination ter-

minals, which is the typical size of the instances to be solved. As this one instance is not enough to properly analyse the characteristics of the algorithm, a generator has been implemented to create additional instances. This generator ensures that problem instances with characteristics similar to the real-world instance are generated. To accomplish this, the following factors have been taken into account:

First, the distribution of transport volumes across the begin terminals and end terminals is determined. Some large begin terminals serve, for example, a high number of end terminals (high connectivity). The same holds true for the end terminals, of which some receive containers from many begin terminals. This is modelled by drawing a random variable from a uniform distribution with range $[0, 1)$ for each begin and end terminal.

Second, the distribution of the origin/destination (O/D) flow matrix is created. As the flows in the real-world O/D matrix are not drawn from a known distribution, bootstrapping is used to generate matrices with a similar distribution as the real-world instance. In the bootstrapping, only the non-zero values are used for determining the distribution, as the distribution of zero-flows is determined by the connectivity of each begin and end terminal. To determine whether the begin and end terminal are connected, the two random numbers of the begin and end terminal are added together to form a modifier in the range $[0, 2)$. The target fraction of flows that should be zero is a parameter to the generator, and set to 0.4 for all instances, as this is approximately the fraction of zeroes in the real-world problem. To determine whether or not a flow should be zero, a random number is drawn from a uniform $[0, 1)$ distribution, and multiplied by the modifier. As the modifier does not have a uniform distribution (adding two uniformly distributed variables yields a triangular shape), a cutoff value is determined such that the chance of a zero is 40%. If the value is designated to be non-zero, a value is drawn using bootstrapping. This value is then scaled with the train size to determine the number of containers to be transported. Repeating this procedure for all O/D pairs yields the required flow matrix.

Last, the distance matrix is generated by determining the distance from each begin terminal to the hub, and from the hub to each end terminal. This is done by generating two weakly increasing list of values between a minimum and a maximum value. The minimum and maximum distance are determined by the day-relations (one-day, two-day, and multi-day distances), which are part of the input. The distance from origin to hub is a quarter of the total distance, the distance from hub to destination is the remaining three-quarters of the total distance. The total distance for each O/D pair can therefore be calculated by adding the origin-hub and hub-destination distances. It is assumed that a small detour of 10 kilometres is incurred when visiting the hub.

After the instance has been generated, it is saved in a format that can be read by the algorithms.

5.1.3 Integer Linear Programming solver

In order to solve the ILP-formulation of the Twin Hub Bundling (THB) problem and evaluate the algorithms, Gurobi [16] was chosen as the ILP-solver. Gurobi 5.0.0 64-bit is a Linear Programming (LP) and Mixed Integer Linear Programming (MILP) solver – Integer Linear Programming (ILP) is a subset of MILP – that is optimized for use on multi-core machines. A Java program is used to read the problem from a file, and create an ILP-model using the Java-interface supplied with Gurobi. The Java program then calls the Gurobi solver with the required parameters, such as maximum runtime and a known start solution if available. Gurobi then uses the branch-and-bound and the branch-cut-and-price algorithm to solve the ILP problem. Also, sophisticated heuristics are used to generate integer solutions from intermediate non-integer solutions, which allow the solver to progress more quickly by allowing it to prune the search tree earlier in the process.

The evaluation of the algorithm will focus on establishing performance characteristics in terms of objective value and runtime in relation to the input parameters, both for the generated instances as well as the real-world instances. Furthermore, we investigate how the choice of a maximum number of batches influences the solution quality reached after a certain runtime. We then investigate whether the asymmetry in the formulation caused by introducing indirect trains has significant influence on the runtime.

5.2 Experimental results

In this section, we will investigate the properties of both the ILP-algorithm and the cascade algorithm, guided by the questions posed at the start of this chapter. Each of the following sections describes an experiment answering one of these questions. We will first introduce the experiment and describe the datasets used, followed by the testing methodology used in that experiment. A discussion of the results concludes each experiment.

5.2.1 What is the performance of the ILP-algorithm on the real-world instance?

While we cannot reasonably expect that the real-world instance can be solved to optimality, due to the exponential nature of the algorithm, good-quality solutions that approach the optimal solution may still be found by the algorithm. We will now investigate how large this ‘gap’ is between the objective value of the best solution found by the algorithm and the calculated lower bound.

Data

Only one instance was provided, which consists of 18 source terminals and 50 destination terminals. However, the problem instance can be decomposed

into subproblems based on the distance between the source and destination terminals and the speed of the trains. We split the instance in three distance relations: the A/B-, A/C-, and A/C-relation, consisting of the terminals that can be reached in one, two, and more than two days respectively. See Section 3.1 for more information on distance relations. By splitting the instance on distance relations, three separate instances are obtained with 22, 17, and 11 end terminals. For this test, all three instances will be used, though we will focus on the instance with A/B relation, as this is the largest instance.

Methodology

The ILP-algorithm is run on each problem instance for 86400 seconds (= 24 hours), during which a progress monitor records the progress in solution quality over time. This way, a trace is produced that shows how the algorithm behaves on each problem instance over time.

Results

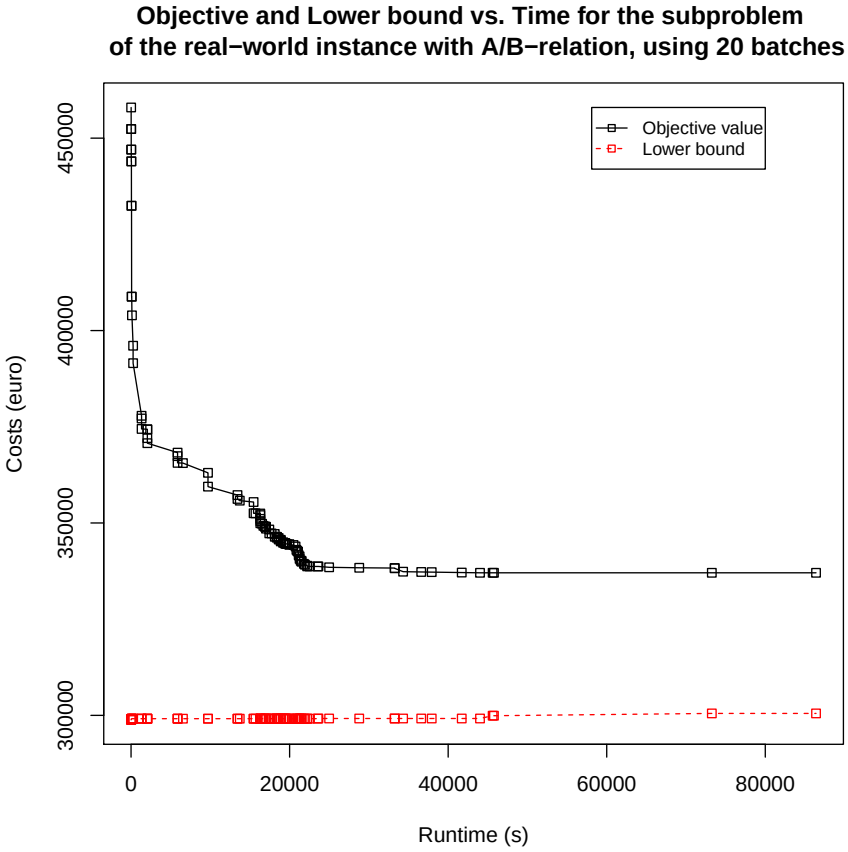

Figure 5.1: Objective value and lower bound for the real-world instance subproblem with ab-relation, using 20 batches. The subproblem is solved using the ILP-algorithm.

Figure 5.1 shows the progress made by the ILP-algorithm in terms of objective value and lower bound on the largest subproblem (the A/B-relation subproblem) for 86400 seconds (= 24 hours). Each data point denotes either an improved solution or an improvement of the lower bound of the problem. The graph shows that progress is quick at first, with more than 90% of the improvement made in the first five hours, and more than 99% in the first eight hours. This does not mean that it is not worth spending more computation resources on finding even better solutions: the solution quality is still improved with time, although not as fast as before. In the last 16 hours, an improvement in performance of another 0.4% is realised. This may not seem like much, but still amounts to a reduction of 1310 euros compared to the solution found after 8 hours. It may still be worth to run the algorithm for more time when computing the solution that will be used in the pilot of the Twin Hub project.

Figure 5.2 shows that the results for the other two subproblems (with A/C and A/D relation) also start out with fast improvement, although the improvement on the A/C-subproblem stops sooner than in the other two (A/B and A/D) subproblems. Progress after 5 to 6 hours is slow for all instances, meaning that it is worthwhile to spend at least 6 hours on solving the problems to obtain a good-quality solution.

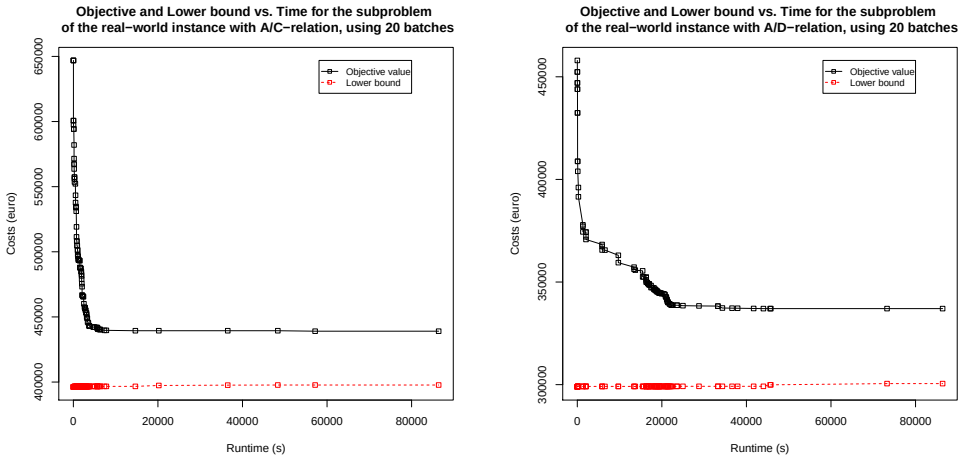

Figure 5.2: Time to reach a certain approximation quality for the real-world instance subproblems with ac- and ad-relation, using 20 batches, solved using the ILP algorithm.

The best solutions in each run resulted in a gap between the objective value and lower bound of 12.2% for the A/B, 10.4% for the A/C, and 7.5% for the A/D subproblem after 24 hours. This is still quite a substantial amount as there might be a solution for the A/B problem costing 36500 euros less than the current solution. While it is not probable that such a reduction can be achieved – there might be no valid solution with that cost – it is not inconceivable that a solution costing several thousands of euros less may exist. It is therefore worth

investigating whether the search can be sped up.

5.2.2 What instances can be solved to optimality within a reasonable time frame by the ILP-algorithm?

As we have seen in the previous experiment, the real-world instance could not be solved to optimality. Previous attempts at solving problems similar to the Twin Hub Bundling problem using Integer Linear Programming, such as the ILP algorithm presented by Newman and Yano [29], have also not been able to prove optimality even for the smallest instances (3 begin terminals and 3 end terminals) within 9000 seconds. In this experiment, we investigate what instances can be solved to optimality using the ILP-algorithm, and what parameters of the problem instance influence the runtime most.

Data

As it is not expected that large problems can be solved to optimality, a test set of 75 small instances was generated using the instance generator. The number of sources and destinations in this dataset was varied between 4 and 12 in increments of 2. For each combination of sources and destinations, three problem instances were generated. For each of the 75 generated instances, the maximum number of batches was varied between 2 and 10 in increments of 2, yielding a total of 375 test configurations.

Methodology

Each test configuration in the dataset is run for a maximum of 1800 seconds using the ILP-algorithm. This maximum will be used to test all small instances throughout this thesis, as some initial experiments showed that increasing the runtime beyond this point did not yield significantly better results. If the algorithm does not finish before the maximum runtime has elapsed, the execution is stopped, and the best result found thus far is returned. For each configuration, the 'gap' is calculated between the objective value (i.e. the costs of the best solution found thus far) and the lower bound calculated by the algorithm. If the solution found is not proven optimal, the gap will be greater than zero. This will show what instances can be solved to optimality.

Results

The results of the runs were averaged for each combination of number of sources, destinations, and batches, yielding 125 data points spread over these three dimensions. Figure 5.3 shows the mean gap through the colour-coding of the data points. The green dots show where the observed gap is zero, meaning that all three instances could be solved to optimality within the maximum runtime of 1800 seconds. The instances that could not be solved to optimality

are colour-coded from yellow to red to show the mean size of the gap for each combination of sources, destinations, and maximum number of batches.

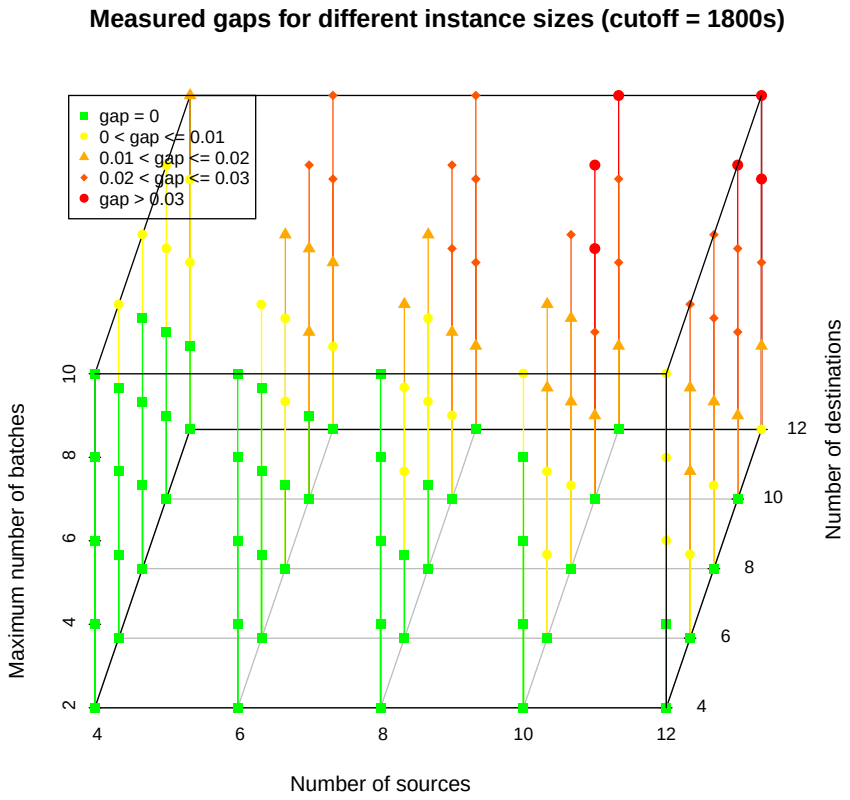

Figure 5.3: Observed gap between the objective value reached and the calculated lower bound. Each point is the mean of the results of three instances run for a maximum of 1800 seconds.

Figure 5.3 confirms that it is hard to find proven optimal solutions for the larger instances of this test set. The largest instances that can be solved to optimality are those with 8 sources, 8 destinations, and a maximum of 4 batches. It does not seem to be the case that one dimension is dominant in determining whether a problem can be solved to optimality. It is the general increase in problem size caused by an increase in one dimension that seems to matter most, which is not particular to one dimension.

Even though more problems can be solved to optimality using more computing power, the exponential nature of the algorithm makes it very unlikely that this algorithm will solve real-world sized instances (i.e. > 15 sources, > 30 destinations and > 30 batches) to optimality within a reasonable time frame in the near future. However, the relatively low differences between the value found and the calculated lower bound means that the results that are not

optimal are close to the optimal value, meaning that we can expect good-quality results for these instances when running these instances for 24 hours.

5.2.3 Does the cascade algorithm perform better on the real-world instance than the ILP-algorithm?

It is expected that the cascade algorithm will perform better than the ILP-algorithm, as the relaxation is used to provide the ILP-algorithm with a better initial solution. In this experiment, we will investigate if this is the case.

Data

The same real-world problem instance was used as in the tests for the ILP-algorithm, which means that we will test on the three separate subproblems with 22, 17, and 11 end terminals. For this test, all three instances will be used, though we will focus on the instance with A/B relation, as this is the largest instance.

Methodology

For each problem instance, the cascade algorithm is run for 86400 seconds (= 24 hours), of which 900 seconds are used for running the algorithm with the relaxation. The solution provided by the relaxation is then used as an initial solution for the ILP-algorithm, which is run for the remaining 85500 seconds. A progress monitor records the progress in solution quality over time. This way, a trace is produced that shows how the algorithm behaves on each problem instance over time, which is then compared to the trace of the ILP-algorithm.

Results

We have seen in the performance analysis of the ILP-algorithm (Figure 5.1) that a sizeable gap remains in the first hours of computation. When a better initial solution is found, the ILP-algorithm can ignore more solutions that have a higher costs than the solution found using the relaxation. Therefore, when solving the problem to optimality, the cascade algorithm is expected to perform better than the ILP-algorithm. As solving to optimality is infeasible for all but the smallest problems, we should focus on the performance of the algorithms for intermediate milestones when comparing these algorithms, for example when a 15%, 10%, or 5% approximation quality is achieved.

Figure 5.4 shows that the cascade algorithm reaches the 25% and 20% approximation quality marks much faster than the ILP-algorithm does, as the algorithm using the relaxation makes a lot of progress in the first minute. The difference at the 20% mark is as large as 3 orders of magnitude, meaning that the relaxation does a formidable job at finding a better start solution quickly. Unfortunately, this effect seems to diminish when the 15% approximation quality mark is reached, although the ILP-algorithm still takes an hour longer to

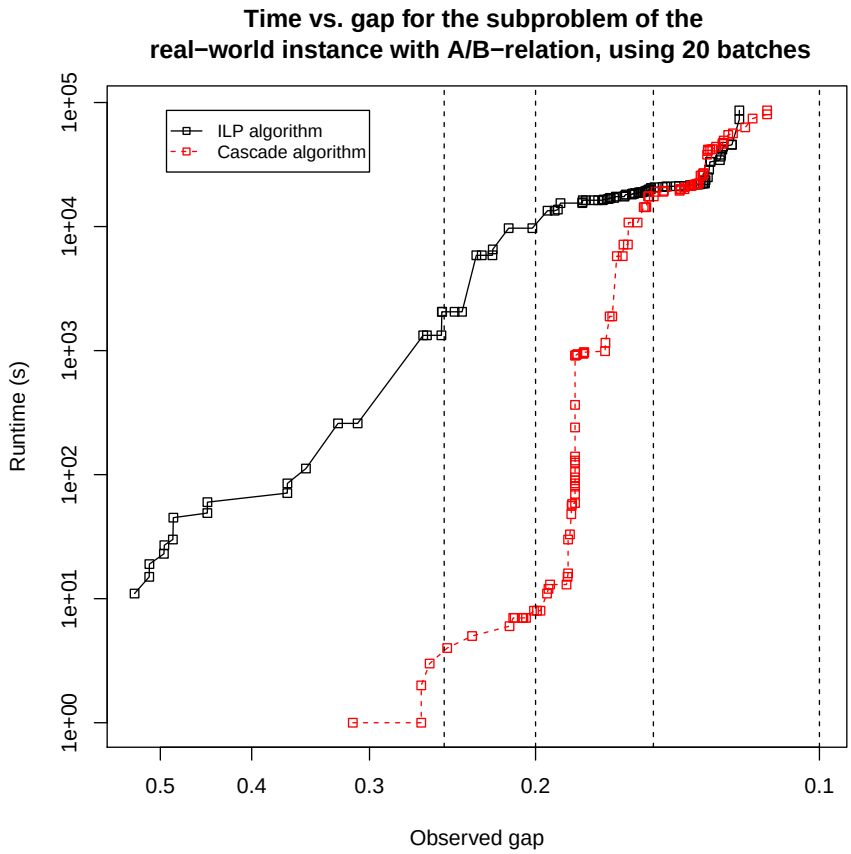

Figure 5.4: Time to reach a certain approximation quality for the real-world instance subproblem with ab-relation, using 20 batches for both the cascade and ILP algorithm.

reach this mark than the cascade algorithm. After 5 hours of runtime, the ILP-algorithm even finds better solutions than the cascade algorithm. This is definitely unexpected as the cascade algorithm uses the ILP-algorithm after the 900 seconds in which the relaxation is run, and therefore starts with a much better solution than the ILP-algorithm starts with. How can the ILP-algorithm be two hours faster to reach 13% approximation quality, then?

Inspection of the intermediate solutions produced by both algorithms shows that the solutions found in the first five hours (where the cascade algorithm is still better) are very different from each other. Table 5.1 shows that the solutions with quality 16.7% are very different from each other, even though they have the same objective value (difference $< 0.01\%$). The most striking difference is the scheduling of indirect trains: in the solution found by the cascade algorithm, only 13 indirect trains are scheduled, while 37 indirect trains are scheduled in the solution found by the ILP-algorithm.

The solutions found after 14 hours by the ILP-algorithm and the cascade algorithm are much more alike. When comparing the solutions with the same

	Solution quality: 16.7%		Solution quality: 12.7%	
	ILP	Cascade	ILP	Cascade
Runtime (s)	16892	1887	44017	46342
Modal split (%)	89.6%	74.1%	90.7%	85.5%
Trains in batches	41	28	33	32
Indirect trains	37	13	29	25
$\frac{\text{Indirect trains}}{\text{Batch trains}}$	90.2%	46.4%	87.9%	78.1%

Table 5.1: Comparison between solutions of the ILP-algorithm and the Cascade-algorithm with the same solution quality.

objective value, the modal split differs only 5.2%, and the number of trains in batches and the number of indirect trains are also quite close to each other. After this, the cascade algorithm performs better.

Hypothesis testing

As the ILP-algorithm itself and the version of the ILP-algorithm included in the cascade algorithm are exactly the same, the observed difference in performance should be sought in the different starting solutions. The default starting solution is one where all containers are transported using direct trains, while the solution produced using the relaxation contains also contains trucks, and hub- and destination trains.

With a better initial solution, parts of the search tree can be cut off earlier, saving computational effort and reducing memory requirements. We assume that this allows the branch-and-cut algorithm used by Gurobi to find the optimal solution faster. However, the same solution is used to provide the initial settings for the variables, which may not actually be advantageous for the solver used by Gurobi, as many variables would need to be changed in order to find a solution with lower costs. When a large number of variables need to be changed in order to find a better solution, we call this solution a 'local optimum'.

The observation of a clear turning point near the 15% approximation quality mark supports this theory. We therefore formulate the following hypothesis:

"The solution produced using the relaxation is in or very close to a local optimum for the ILP-algorithm"

If this hypothesis were to hold, it would offer an explanation for the behaviour of the ILP-algorithm, which makes little progress during the first hours when using the relaxation to find an initial solution. To find evidence concerning whether or not this hypothesis is true, we will formulate a prediction that should hold when the above hypothesis is true:

Prediction: There is a drastic change ($> 25\%$) in the ratio of indirect trains and other batch trains in the produced schedules before and after the turning point. One would expect that, if the solver is stuck in a local optimum, the difference between the initial solution and the solution found just before the turning point do not differ by much, and that the solution just after the turning point is drastically different from the solutions before it. This would also mean that the number of indirect trains in the solution before the turning point would be relatively low, as there are no indirect trains in the initial solution. However, upon inspection of the solutions between the initial solution and the turning point at 15% solution quality, we observe that the transition between solutions with hub- and destination trains to solutions with more indirect trains is gradual: the number of additional indirect trains from one solution to the next is never more than four. Furthermore, the difference in the ratio of indirect trains compared to the total amount of batch trains between the solutions at the turning point is no greater than the differences observed before the turning point. This is inconsistent with the hypothesis, which means we should reject it.

Is the cascade algorithm faster?

Currently, there is no solid explanation for the observed behaviour in Figure 5.4. However, there is some evidence that the observed behaviour is not merely a statistical fluke. Figure 5.5 shows the same type of graph for the two other subproblems, which are smaller. Both exhibit the same kind of behaviour with three typical parts. First, the algorithm using the relaxation finds a solution that is of a much higher quality than the solution found by the ILP-algorithm in the same time. Then, the progress of the cascade algorithm is slow, allowing the ILP-algorithm to catch up and find better solutions in the same amount of time. In the third part, the ILP-algorithm struggles to find better solutions, while the cascade algorithm continues to make progress, overtaking the ILP-algorithm again.

An intuition for the observed behaviour in the third part is that the cascade algorithm takes advantage of the parts of the search space that could be pruned using the higher-quality initial solution. This relies on the assumption made earlier that a better-quality initial solution causes a speed-up in finding the optimal solution. This assumption could be tested by generating a sufficient amount of instances using the instance generator, and running the cascade algorithm twice on each instance: one time when using the relaxation to produce a solution with a 'gap' of 5%, and once when running the algorithm using the relaxation for the full 900 seconds. In both cases, the ILP-part of the cascade algorithm should be run for the same amount of time (24 hours minus 900 seconds). The hypothesis test with a one-sided alternative hypothesis that the cascade algorithm with the relaxation algorithm run for the full 900 seconds produces better results (i.e. has a smaller gap and therefore a better approximation quality) can then be evaluated using the Mann-Whitney U-test. It is

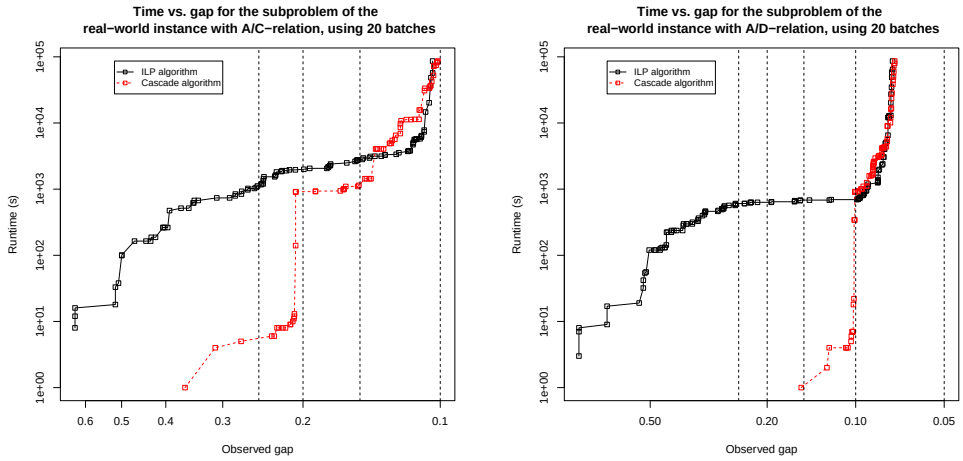

Figure 5.5: Time to reach a certain approximation quality for the real-world instance subproblems with ac- and ad-relation, using 20 batches for both the cascade and ILP algorithm.

expected that the cascade algorithm with the better initial solution produces results that are significantly better. Due to a lack of time, this test could not be run, as the expected difference in solution quality is low, requiring a large amount of instances to show whether the expected effect exists.

In conclusion, it depends on the size of the problem instance in combination with the amount of time available for solving the problem whether or not the cascade algorithm is faster or not. At both extremes: when the algorithm can be run for only a short time, or for a relatively long time, the cascade algorithm produces better results. When run for an intermediate amount of time, the ILP-algorithm can produce better results, although the difference with the cascade algorithm never gets as large compared to the gap in solution quality achieved by the relaxation algorithm in the first 900 seconds.

5.2.4 Does the cascade algorithm solve larger instances to optimality than the ILP-algorithm in the same time?

The cascade algorithm uses a relaxation of the problem to provide the ILP-algorithm with a better initial solution. This is expected to cause a speed-up in the algorithm.

Data

The same 375 test configurations that were used to evaluate the ILP-algorithm will be used to evaluate the cascade algorithm, making the results of both algorithms comparable.

Methodology

Each test configuration in the dataset is run for a maximum of 180 seconds using the relaxation of the problem, and then 1620 seconds using the ILP-algorithm. It was observed in some initial experiments that 180 seconds is sufficient to solve the relaxation to optimality, or at least come within 1% of the optimal solution. If the optimal solution of the relaxation is found, the ILP-algorithm is still run for 1620 seconds, which may cause the total runtime to be slightly less than 1800 seconds. If the relaxation or the ILP-algorithm does not finish before its maximum runtime has elapsed, the execution is stopped like in the test of the ILP-algorithm. The gap after 1800 seconds shows what instances could be solved to optimality, and, if these could not be solved, how close to optimality the algorithm could get.

Results

The results of the runs were averaged for each combination of number of sources, destinations, and batches, yielding 125 data points spread over these three dimensions. Figure 5.6 shows the mean gap through the colour-coding of the data points. The green dots show where the observed gap is zero, meaning that all three instances could be solved to optimality within the maximum runtime of 180 + 1620 seconds. The instances that could not be solved to optimality are colour-coded from yellow to red to show the mean size of the gap for each combination of sources, destinations, and maximum number of batches.

Figure 5.6 shows that the cascade algorithm does not seem to be significantly better than the ILP-algorithm. If we compare this result with Figure 5.3, the results differ only in one data point (10 sources, 8 destinations, 8 batches). This means that the difference is too small to show up in this figure, so a more detailed investigation is required to see whether there is any difference at all.

Figure 5.7 shows a comparison between the performance of the cascade algorithm and the ILP-algorithm: The blue-coloured dots represent the instances where the cascade algorithm performed slightly better, and the yellow-coloured dots the instances where the cascade algorithm performed slightly worse than the ILP-algorithm. This figure shows that there is no clear performance gain for a particular subset of the instances, although for larger problem instances (10-12 sources, 10-12 destinations, 8-10 batches), the cascade algorithm performs better than the ILP-algorithm.

Figure 5.8 shows that in most cases, there is no difference between the best solution found by the cascade algorithm and the ILP-algorithm. In the cases that there is a difference, there seems to be almost as many cases for which the cascade algorithm performs worse than for where it performs better. The mean of the data set is -10.9, meaning that the cascade algorithm performed slightly better, although the the sample standard deviation is 148, meaning that we should perform a test for significance. The Mann-Whitney U-test with

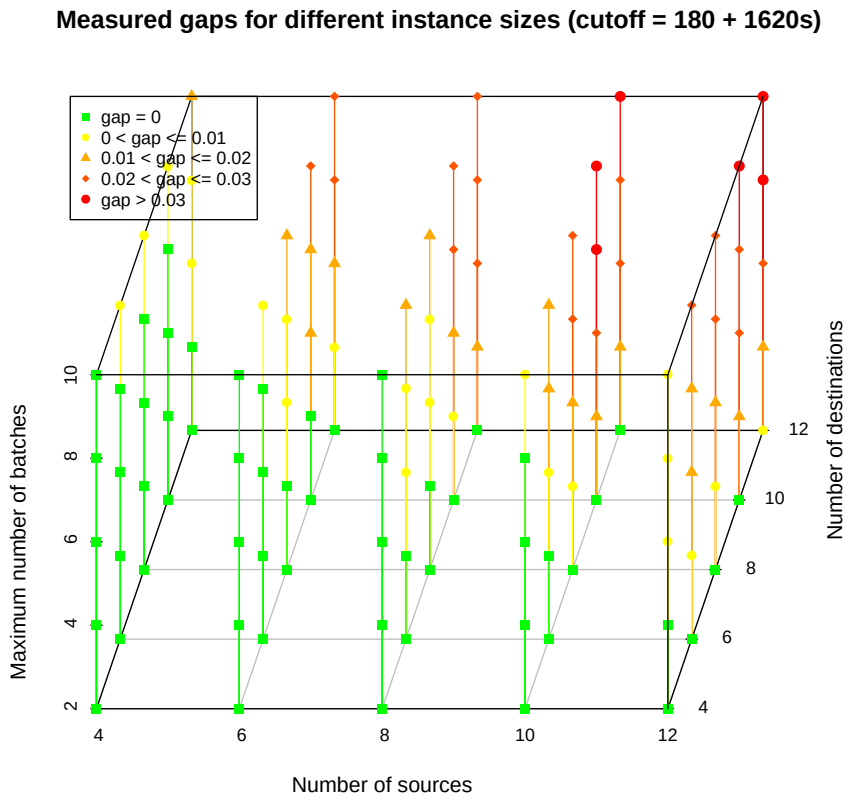

Figure 5.6: Observed gap between the objective value reached and the calculated lower bound. Each point is the mean of the results of three instances run for a maximum of 180 seconds on the relaxation and 1620 on the ILP-algorithm.

an alternative hypothesis that the best objective value found by the cascade-algorithm is lower (i.e. better) than the best objective value found by the ILP-algorithm yields $p = 0.2104$, meaning that we cannot conclude that the cascade algorithm performs better than the ILP-algorithm based on this data set.

This result was expected, as for smaller instances both the cascade algorithm and the ILP-algorithm are likely to find the best solution. In the cases where the cascade algorithm performed worse, the ILP-algorithm found a better solution than could be found using the relaxation in the first 180 seconds, which is not uncommon for these small instances as the best solution that can be reached using the relaxation has a higher objective value than the best solution that can be found by the ILP-algorithm, as the relaxation does not consider indirect trains. Only for larger instances, where the ILP-algorithm cannot find a good-quality solution quickly, the relaxation will provide a good-quality solution faster than the ILP-algorithm can, as we have seen when solving the real-world instance. By providing the ILP-algorithm with this solution, this

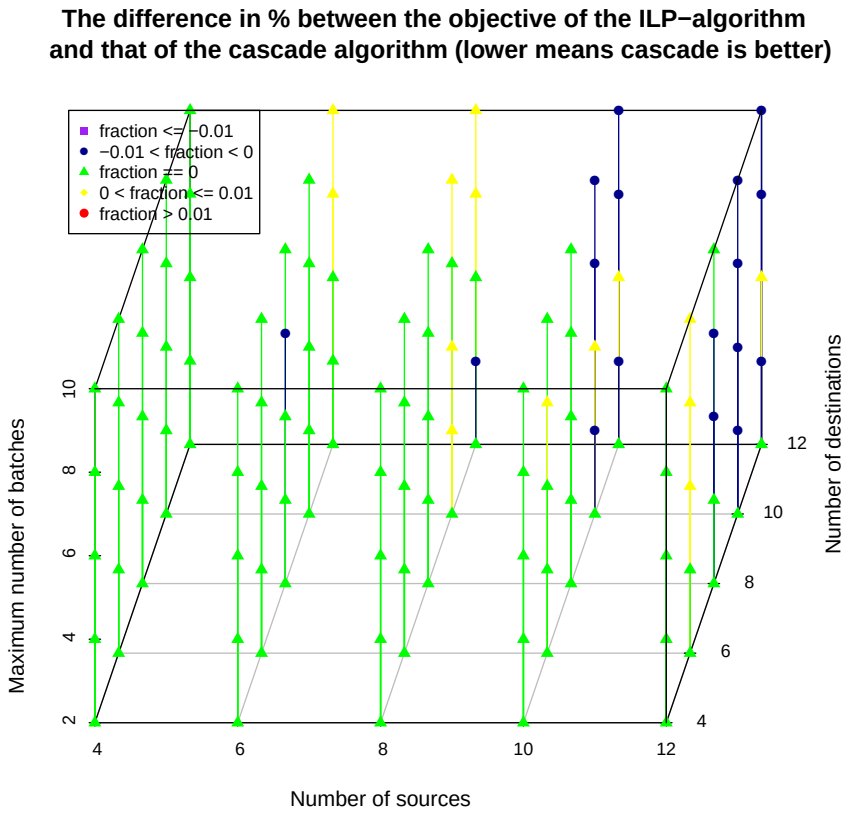

Figure 5.7: Difference in observed gap between running the full ILP-algorithm for 1800 seconds and first running the algorithm using the relaxation for 180 seconds, and then running the ILP-algorithm for another 1620 seconds.

algorithm can get a head start and yield a better solution after the same amount of time.

5.2.5 Does the asymmetry in the formulation of indirect trains have a significant influence on the runtime?

When looking at the number of variables and constraints used to specify the problem, the specification of indirect trains contributes most to the size of the model. This is because, unlike the trucks, direct trains, hub trains, and destination trains, the containers being transported on an indirect train are either (un)loaded at the hub, or stay on the train for the whole journey. Differentiating between these containers is required for the correct accounting of the transfer costs, as no transfer costs are incurred for containers that stay on the train. For each indirect train connection – described uniquely by its source terminal, destination terminal, and batch number – an additional variable is needed to describe the destination of the container. This means that four vari-

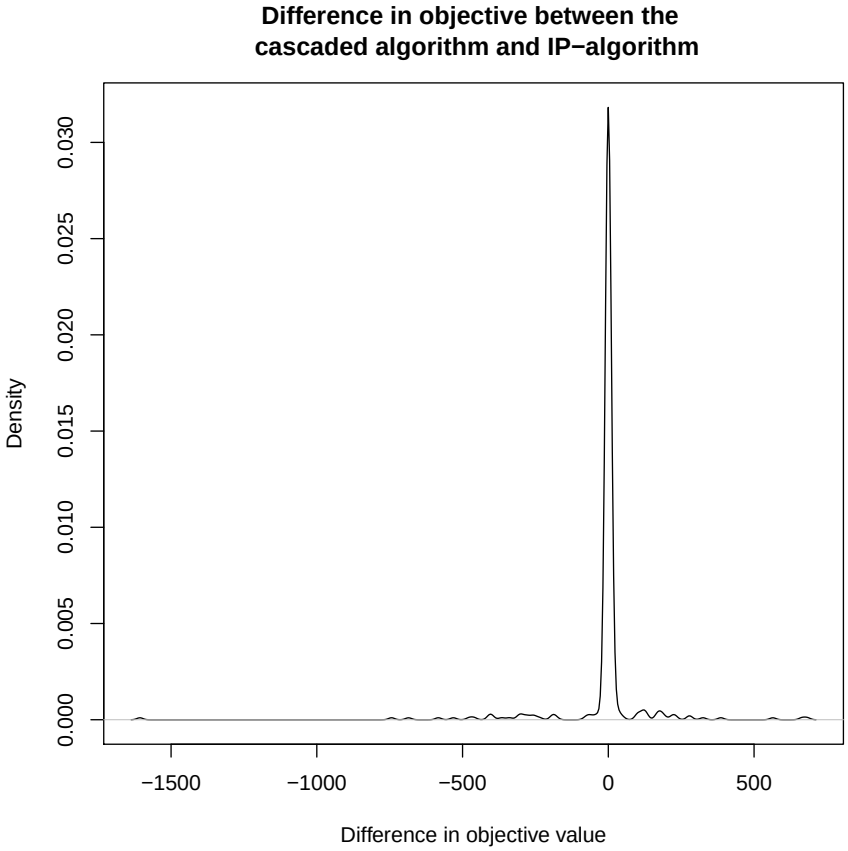

Figure 5.8: Kernel density plot of the difference in objective value between running the ILP-algorithm for 1800 seconds and first running the algorithm using the relaxation for 180 seconds, and then running the ILP-algorithm for another 1620 seconds (cascade algorithm). A lower value means that the cascade algorithm is better.

ables are needed to uniquely describe a transfer for an indirect train, of which two denote a destination terminal.

The size of the model is, therefore, more dependent on the number of destinations (quadratic) than on the number of sources (linear). This asymmetry in the model might have an influence on the performance of the algorithm. We therefore devise the following test:

Data

Two sets of 35 instances are generated: set 1 containing problems with 6 sources and 12 destinations, and set 2 containing the same problem instances as set 1, but with the sources and destinations switched, meaning that these problems contain 12 sources and 6 destinations.

Methodology

To test whether the asymmetry in the formulation has a significant influence on the performance of the algorithm, we formulate the following pair of hypotheses:

> H_0 : A problem with m sources and n destinations $(m < n)$ will have an objective value **equal to** the mirrored problem with n sources and m destinations when run for a certain amount of time and optimality has not been reached.
>
> H_1 : A problem with m sources and n destinations $(m < n)$ will have an objective value **greater than** the mirrored problem with n sources and m destinations when run for a certain amount of time and optimality has not been reached.

The hypothesis is tested by running the ILP-algorithm for 1800 on each instance in the two sets, using a maximum of 8 batches. The anticipated result is that the best solution found for a problem with 6 sources and 12 destinations will have a higher objective value than the mirrored problems with 12 sources and 6 destinations, as the model for the first set (6 sources, 12 destinations) is much larger. For this test, we choose a significance level $\alpha = 0.05$, meaning that we reject H_0 in favour of H_1 when the p-value of the Mann-Whitney U-test is lower than 0.05.

Result

Figure 5.9 shows the difference in objective value between set 1 and set 2. It is clear that the left tail is much larger than the right tail, with the instances in set 1 having a 0.05% (103 euros) larger objective value than the instances in set 2. The test yields p-value of 0.020, which is lower than 0.05. This means we reject H_0 in favour of H_1, and conclude that the quadratic number of variables used to encode the transfers of indirect trains has a significant, though relatively small influence on the objective value reached after a certain amount of time.

The conclusion that can be drawn is that better results may be obtained when the instance is mirrored (i.e. the source and destination terminals are swapped) if the number of destination terminals is significantly larger than the number of source terminals.

This test has also inspired the creation of the relaxation of the problem formulation, where the quadratic part of the formulation (i.e. the indirect trains) is left out. The relaxation is presented in Section 4.2.

5.2.6 What is the influence of the choice of a maximum number of batches on the performance?

The maximum number of batches is not a parameter that follows directly from the problem. There may be a limit on the number of trains that can be serviced

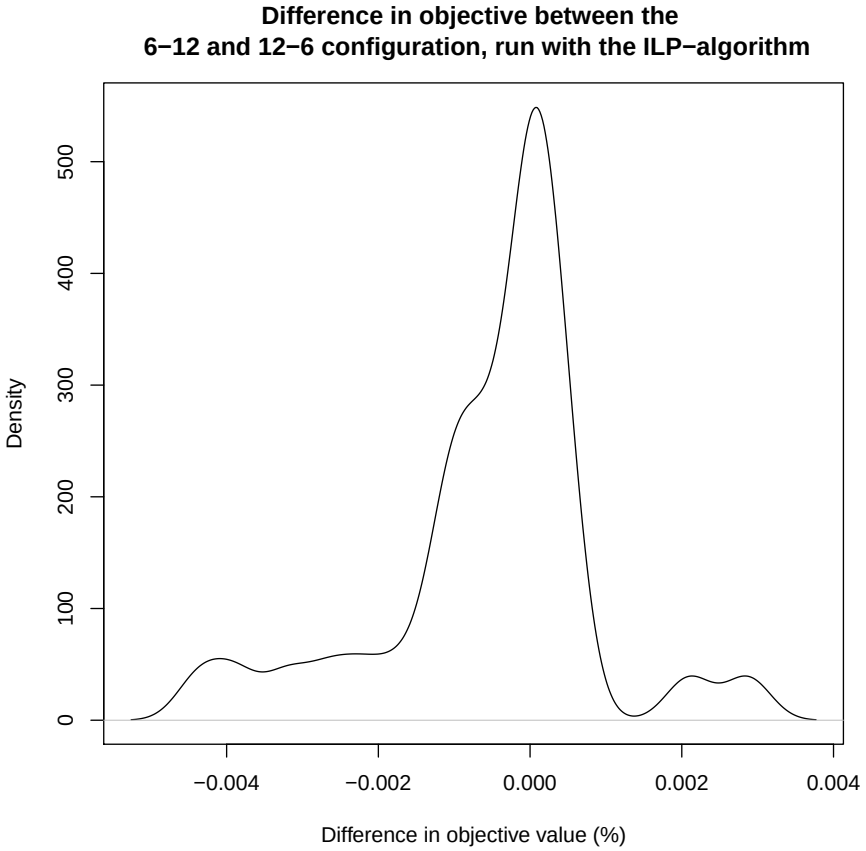

Figure 5.9: Mean observed gap when running the two test sets for 1800 seconds

at the hub at any one time, but this limitation is ignored when calculating the possible cost benefits of using a Twin-Hub network. The maximum number of batches should, therefore, be chosen in such a way that it is not a limiting factor. However, choosing this number higher than the number of batches in the optimal solution increases the model size, as the number of variables and the number of constraints are linear in the maximum number of batches. We will investigate whether choosing the maximum number of batches too high has a significant influence on the performance below.

Data

Two datasets will be used: dataset 1 includes three instances with 12 sources and 12 destinations, which are selected from the larger dataset of 75 instances that was generated for the earlier performance tests. Nine settings for the maximum number of batches are used for each of these three instances: from 2 to 10 in increments of 2, and from 10 to 30 in increments of 5. Dataset 2 contains 17 generated instances with 12 sources and 12 destinations. Here, only two settings are used for each of these instances: 6 batches and 30 batches.

Methodology

First, the 27 configurations in dataset 1 are run for 1800 seconds to investigate the effect of increasing the maximum number of batches in more detail. The objective value reached after 450, 900, and 1350 are also recorded to test if the optimal choice of the number of batches varies with runtime. Then, the instances of dataset 1 and 2 combined are tested with 6 and 30 batches to test whether there is a significant difference between the objective value found after 1800 seconds. For this purpose, the Mann-Whitney U-test will be used with the null-hypothesis that there is no difference between the objective values, and the alternative hypothesis that the objective value found with 6 batches is smaller than that found with 30 batches. A significance level of 0.05 is chosen, and it is expected that the difference is significant.

Results

Figure 5.10 shows that increasing the number of batches has a positive effect on the objective value up to 6 batches when running on the instances in dataset 1. The objective values shown are the average of the three instances. When the maximum number of batches to be used by the algorithm is increased further, however, the objective value of the best solution found is higher than the value found for a lower maximum, as the model size is increased while these extra options are not used in the solution.

Choosing the maximum number of batches for a certain problem instance is, therefore, not a straightforward task. The optimal choice not only depends on the number of batches in the optimal solution, but also on the amount of computing resources at hand. Using more computing resources (i.e. a more powerful computer or letting the algorithm run for a longer period of time) shifts the optimal choice of the number of batches closer to the number of batches used by the optimal solution. Figure 5.11 shows that when running the algorithm for only 450 seconds, the best solution for a problem with 12 sources and 12 destination is found for 8 batches, while for 1350 and 1800 seconds, the solution found with 6 batches has a lower objective value than the solution with 8 batches. When even more computing resources are used, the objective value of the solution found with 8 batches will be equal to or even better than the solution found with 6 batches: a solution with a maximum of 6 batches is also a solution with a maximum of 7, 8 or more batches. As these additional computing resources spent on computing the same solution found with a lower number of batches are essentially wasted, it is important that the number of batches is chosen carefully.

For each problem instance and runtime, there is a different optimum for the choice of the maximum number of batches to be used in the problem. To the left or to the right of the optimal setting, a worse objective value will be found when the runtime has elapsed. As a guideline for choosing the maximum number of batches, however, it is better to err on the low side than choosing

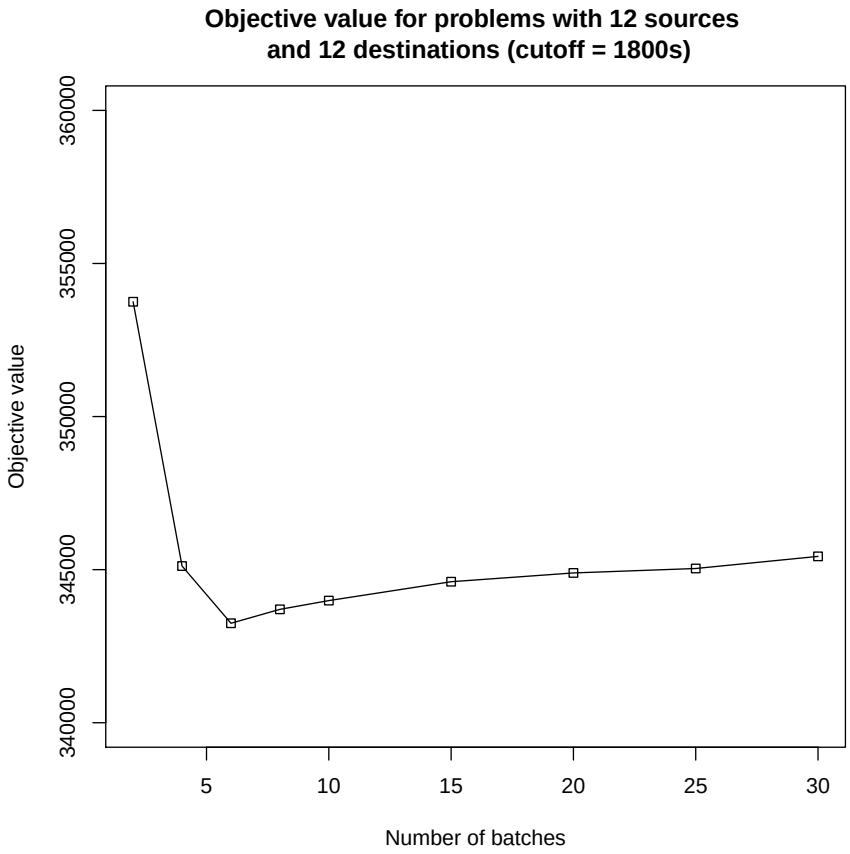

Figure 5.10: Objective value for three instances with 12 sources and 12 destinations, varying the number of batches, running for 1800 seconds

a too conservative (i.e. high) maximum number of batches, as the model with less batches will be smaller and therefore solved more easily. The solution to the smaller problem can then be used as a start solution for the larger problem if inspection of the solution shows that the maximum number of batches was a limiting factor for the solution quality, so the time spent on computing the smaller solution is not wasted.

Finally, the significance of the observed effect is tested using the Mann-Whitney U-test. The test is executed on the 20 instances, yielding a mean objective value of 363153 for the tests with a maximum of 6 batches, and a mean objective value of 364460 for the tests with a maximum of 30 batches. The mean difference is, therefore, -1307, with a standard deviation of 1845. The test yields a p-value of 0.00136, which leads us to reject the null-hypothesis (significance level 0.05) in favour of the alternative hypothesis that the runs using a maximum of 6 batches produce a significantly lower objective value than the runs using a maximum of 30 batches. Therefore, the maximum number of batches should not be chosen too high, as this will lead to results that are

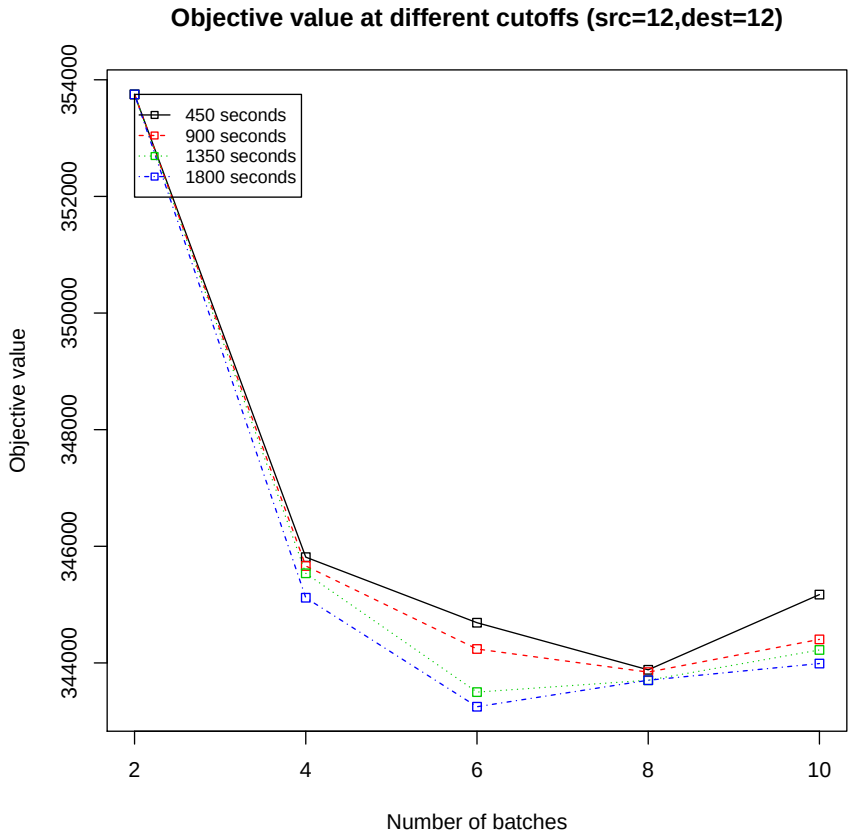

Figure 5.11: Objective value for three instances with 12 sources and 12 destinations, varying the number of batches and running the problems for different lengths of time

significantly worse.

5.3 Discussion

The evaluation has shown that the ILP algorithm produces good-quality solutions to the real-world instance within the required time limit of 24 hours. Tests show that, when running the algorithm for just 30 minutes, finding the optimal solution is only possible for instances smaller than 8 source terminals and 8 destination terminals, with at most 4 batches. It is not expected that significantly larger instances can be solved within 24 hours.

Running the algorithm with the relaxation of the problem formulation in combination with the ILP-algorithm has yielded slightly better results in some cases, but slightly worse results in other cases. The difference in solution quality between running the ILP-algorithm with and without the better initial solution was not significant for the smaller testcases. Nevertheless, the relaxation does prove its usefulness when running large problems for a relat-

ively small time, with better results for the real-world problems being found much faster than with the ILP-algorithm alone. The results show that both the ILP-algorithm and the combination of the relaxation and the ILP-algorithm (cascade algorithm) meet the performance requirements, meaning that both algorithms are included in the bundling tool.

Chapter 6

Bundling Tool

The goal of this thesis is developing a bundling tool that creates Twin Hub Bundling (THB) networks based on the projected origin/destination (O/D) flows in a certain transportation network. In the literature survey (Chapter 2), we concluded that little research has been done on this type of bundling, and that a new algorithm had to be developed. In Chapter 4 we have presented two algorithms for solving the THB problem, which were tested and found to be sufficient for including them in the bundling tool in Chapter 5. Now we have arrived at the point where the bundling tool itself can be created.

The requirements for the bundling tool were described in Section 1.3, where three types of requirements were distinguished: control requirements that describe what parameters of the Twin Hub Bundling (THB) problem should be controllable from the bundling tool, user interface (UI) requirements that specify the interaction with the tool, and performance requirements that describe the minimum performance of the algorithms that form the core of the bundling tool. These requirements specify that the tool must contain three features: a way of specifying the problem instances in the tool, a means of running one of the algorithms on the problem instance, and a way of viewing the resulting THB networks. Each of these features are described in more detail in the following sections.

6.1 Specifying the problem instance

The instance of the Twin Hub Bundling (THB) problem needs to be specified in the bundling tool in order to compute the best THB networks for that problem. This is done in the 'Run Configuration' screen which is the start screen of the tool. Figure 6.1 shows the initial state of this screen, in which the standard settings such as the costs and capacities are set to their default values.

At the top of the screen, there are two fields where the path to the origin/destination (O/D) matrix can be entered. These matrices are usually fixed for each problem, as the O/D matrix is the result of an earlier modal shift analysis and the lengths of the routes between the terminals are fixed. Note that the supplied

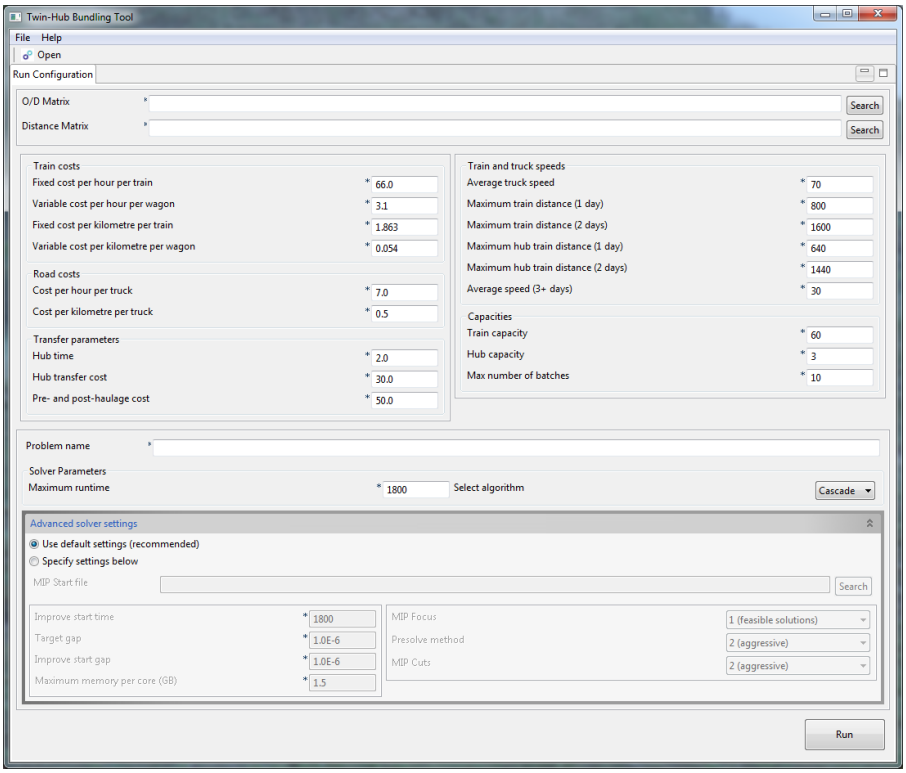

Figure 6.1: The 'Run Configuration' screen of the bundling tool, in which the THB problem instance can be specified.

O/D matrix is assumed to only contain the flows that can be bundled together, so flows having a common distance-relation (A/B, A/C, or A/D), and using the same hub (Rotterdam or Antwerp).

Below the matrix input fields, there are fields to enter the cost parameters and several other parameters that are needed to compute a solution to the problem, such as the distance a train can travel in one day, the capacity of a train, and the maximum number of batches that can be used to solve the problem. These input fields meet the control requirements stated in Section 1.3.2. When these parameters are all entered, the problem instance has been specified completely.

In the third section of the screen, the detailed parameters of the bundling algorithm can be specified. In the first input field an identifier for the problem instance can be provided for future reference. The other two required parameters are the maximum runtime for which the algorithm can be run, and an option for choosing one of the two available algorithms. The cascade algorithm is the default option, as this algorithm performs much better in the first hours of computation for real-world instances (see Section 5.2.3).

Additionally, there is a drop-down box containing advanced solver settings that do not necessarily have to be filled out. These options include selecting

a so-called MIP Start file that is produced when the solver reaches its maximum runtime or is aborted, and includes the information required to resume calculation on the same problem instance. Note that different cost and capacity values may be chosen when the computation is restarted, but it is advised that the capacities are only increased compared to the previous run, as the previous solutions are then also valid for the new run. Other options that can be adjusted are detailed settings for the solver that need not be changed, but allow for more control over the Gurobi solver. These settings are explained in the Gurobi manual that can be found on the Gurobi website [17].

After the instance is specified and the solver parameters are set, the 'Run' button can be clicked to start the solver. When this button is clicked, a new tab is opened showing the progress of the solver, which is explained in the next section.

6.2 Running the algorithm

When the run has been specified and the 'Run' button has been clicked, the 'Progress' screen is opened showing information about the solver. Figure 6.2 shows this screen running the cascade algorithm on one of the real-world sized problem instances.

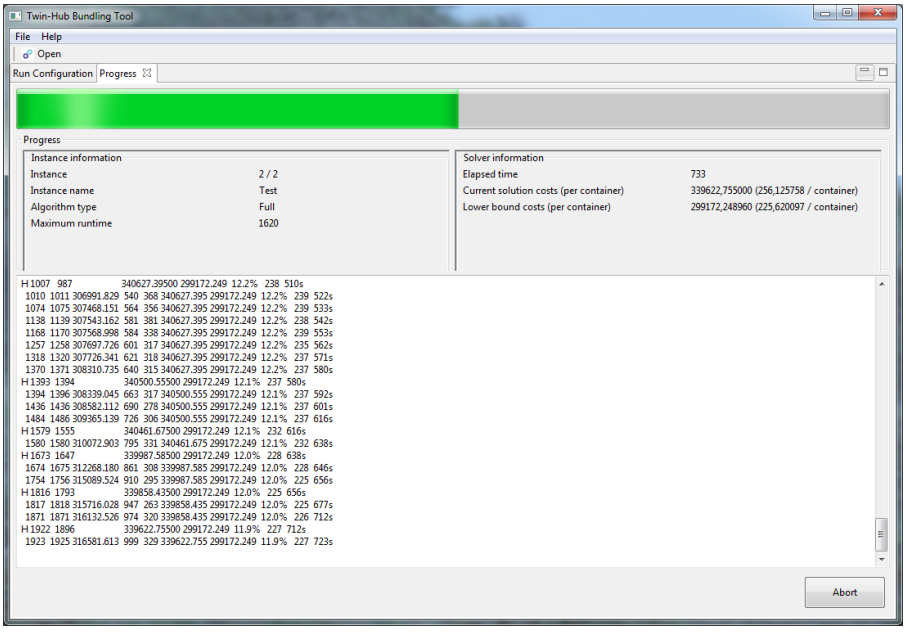

Figure 6.2: The 'Progress' screen of the bundling tool, which shows the progress of the solver.

The progress screen has three sections that together show the progress the algorithm is making on solving the problem. At the top of the screen there is a large progress bar showing what fraction of the total runtime has already

elapsed. This progress bar is only dependent on the runtime of the solver: it is not an indication of the quality of the solution found.

The middle part of the screen does show this information. It shows in which part the solver currently is – running the relaxation or the IP algorithm – and shows how many seconds have already elapsed since this algorithm has been started. In the right part of the middle screen, there are also two key statistics of the best solution found so far: the total costs and the calculated lower bound for the solution. This lower bound is the minimum costs that any solution to the Twin Hub Bundling problem can have. This does not mean that there actually exists such a solution, as not all requirements to the solution might be met by a solution with those costs. Over time, this lower bound may get higher as more possible solutions are investigated by the algorithm. When the lower bound is equal to the best solution found, this means that the solution is optimal and no better solution can be found for that problem.

The third part shows the output of Gurobi – the solver that is used by the algorithm of the bundling tool. This output can be used for diagnostic purposes, as this provides low-level information about the progress of the solver.

Finally, the screen contains an 'Abort' button with which the solver can be stopped. When aborting during the execution of the IP-algorithm, the tool aborts the solver and starts the process that would otherwise be executed when the maximum runtime is reached: the solutions found thus far are saved, and the best solution is displayed in a new tab. This tab is discussed in the next section.

6.3 Viewing the results

When the solver is done, the solutions that have been found by the solver are saved in the working directory. The name of these solution files contain the name of the problem instance that has been entered in the Run Configuration screen and an index. The lower the index, the lower the costs of that solution. The solution with index 0 therefore contains the best solution found. This solution is opened automatically when the solver is done, although the solutions can also be opened later by clicking the 'Open' button in the toolbar or selecting 'Open' from the 'File'-menu.

6.3.1 Viewing the solution

Figure 6.3 shows the screen displaying a solution that is the result of running the cascade algorithm on one of the real-world sized problem instances. At the top of the screen statistics about the entire solution are displayed. This includes information about the problem, such as the number of terminals, O/D flows, and the number of containers to be transported, as well as statistics about the solution itself, for example costs, the exchange of containers, the modal split,

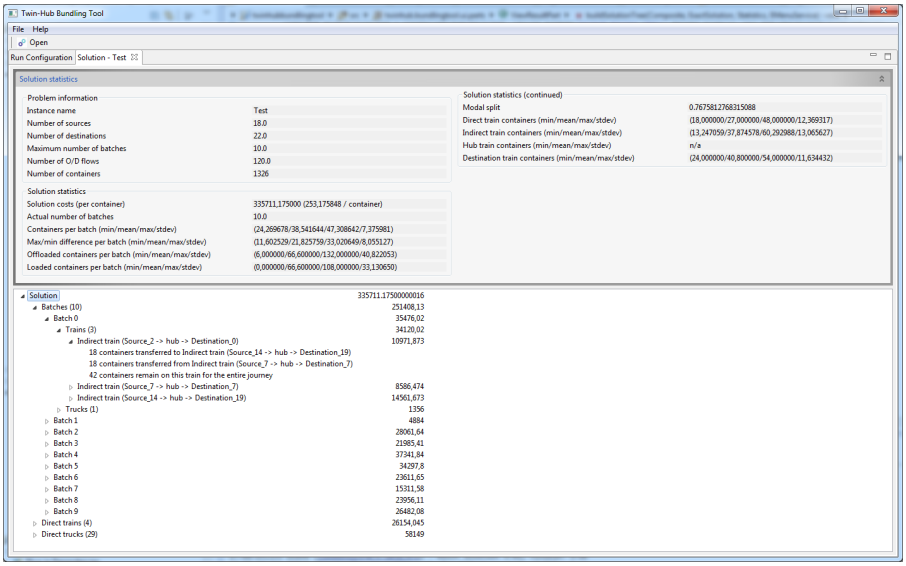

Figure 6.3: The 'View Result' screen of the bundling tool, which shows the solution and statistics about that solution.

and the types of trains used and the average number of containers transported on them.

The bottom half of the screen shows the solution in more detail. The solution is displayed as a tree, where for each item, the cumulative costs for each sub-item is displayed. For example, the number next to 'batches' represents the cumulative costs of all batches. Similarly, the costs next to each batch represent the costs of all trains and trucks in that batch, etcetera. For each train or truck, the containers loaded and unloaded during its journey are displayed. This way, a full overview of all the trains and trucks in the system and the containers transported on them is presented. However, this screen only shows statistics about the entire solution, while one of the requirements was that statistics should also be provided for each train or truck in a batch. This information can be viewed in the 'Batch detail' screen, which can be opened by selecting a batch, and selecting 'Show detailed info' from the right-click context menu (see Figure 6.4).

6.3.2 Viewing details of a batch

Figure 6.5 shows the 'Batch detail' screen, where detailed statistics about a batch and the trains and trucks belonging to that batch are shown. At the top of the screen, the statistics about that batch are shown, such as the costs of the trains and trucks in that batch, and the number of containers that are transported in that batch. Note that the number of transported containers may be a fraction, as transporting a container to or from the hub only counts as a fraction of the total journey of that container. Adding all these fractions together always yields

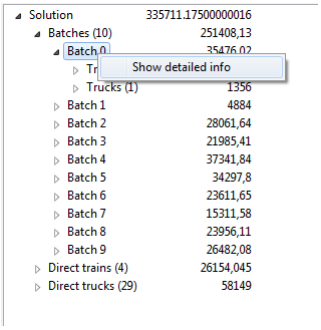

Figure 6.4: Opening detailed statistics about a batch in the 'View Result' screen

the total number of transported containers in a solution.

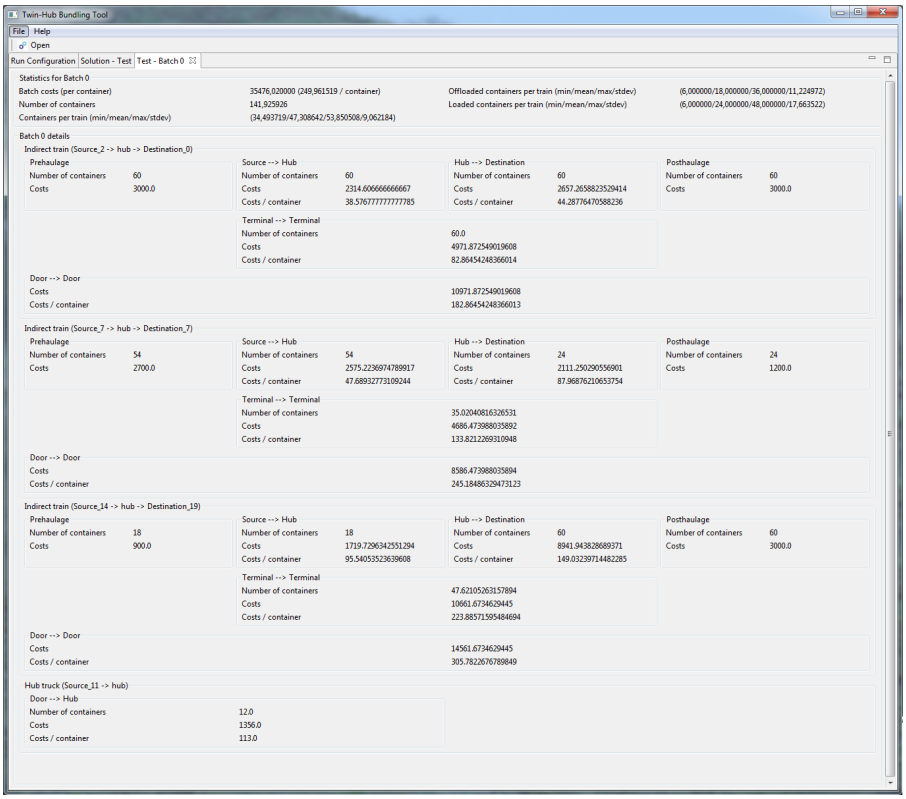

Figure 6.5: The 'Batch detail' screen of the bundling tool, which shows detailed statistics about the trains and trucks in a batch.

Below the statistics of the batch, the statistics of each train and truck in that batch are shown. For each train, the costs are divided in four parts: the prehaulage costs, the costs of transporting the containers from the source terminal of the train to the hub, the transportation costs from the hub to the destination terminal, and the post-haulage costs. For each of these groups, the number of containers, the cost per container, and the total costs are shown. Underneath

these groups, the costs are aggregated to show terminal-to-terminal costs and door-to-door costs for the containers travelling on that particular train. For trucks, the cost structure is simpler, as the pre- or post-haulage costs are included in the journey to or from the hub.

6.3.3 Comparing solutions

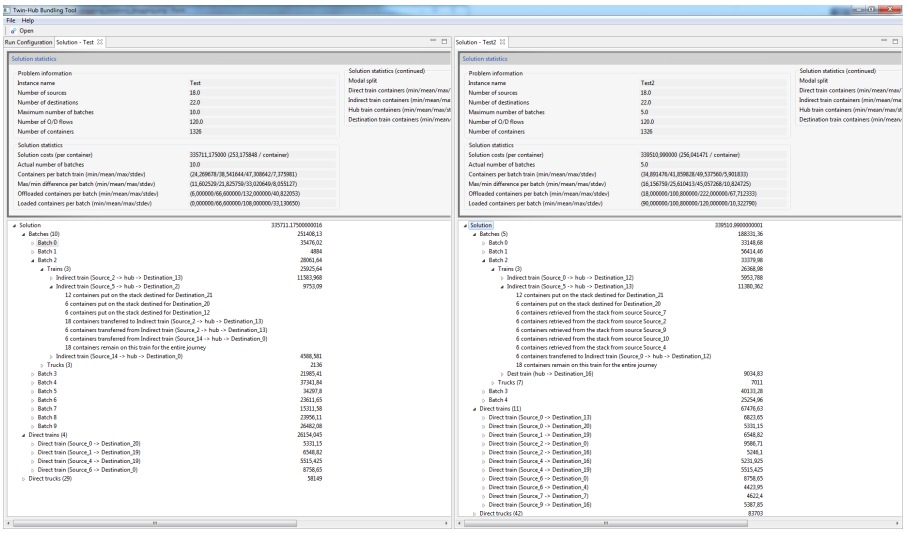

Figure 6.6: The tool allows for segmentation of the tool window, allowing to view multiple tabs at once. This can aid in comparing different solutions that have been found by the algorithm.

The tab-based structure of the bundling tool allows for dynamic reordering, meaning that by selecting one of the tab headers and dragging them to the side of the screen, the screen of the bundling tool will be partitioned in two (or more) sections, each containing a different tab. Figure 6.7 shows the result of opening two solutions, then dragging the tab of one of the solutions towards the edge of the tool window and releasing the left mouse button once the outline of the screen shows the desired partition of the window. The two solutions can now be viewed side by side, allowing for easier comparison between two solutions, batches, or a combination of these.

Another way of viewing multiple solutions or batches at once is to drag the tab outside of the tool window. This creates a new window which in turn can contain multiple tabs. This window can then be dragged onto a second display, from where it can be accessed as usual. Each created window can itself be partitioned in multiple partitions, making the tool very flexible in terms of layout, optimizing the interaction with the tool.

Figure 6.7: Tabs can be dragged outside of the tool window, creating a new window that shows another tab, which can be used to compare different solutions or batches.

6.4 Discussion

The bundling tool presented in this chapter meets all requirements that have been specified in Section 1.3. It includes both the ILP-algorithm and the cascade algorithm for solving the THB-problem, which have been evaluated to meet the performance requirements in Chapter 5. The bundling tool provides a comprehensive interface to use these algorithms, allowing for easy evaluation of different problem instances and the effect of different parameter settings for these instances, meeting the control requirements. After the algorithm has finished, the tool shows the best result found by the algorithm, and allows the user to view and compare this solution with intermediate solutions found in the same run, or solutions found using different parameter settings. Statistics are presented with each solution to compare two solutions more quickly, and the detailed view of each batch aids in selecting promising Twin Hub Networks. The user interface therefore meets and exceeds the set requirements by presenting a flexible interface for specifying a problem, running the algorithm, and evaluating the results.

Chapter 7

Conclusion and Future Work

In this chapter, we will present the conclusion including the answers to the research questions posed in Section 1.4, after which the the contributions of this work to the field of transport research are highlighted. Finally, directions for future research will be given.

7.1 Conclusion

The goal of this thesis was to create a bundling tool for the Twin Hub project, with which promising Twin Hub Bundling (THB) networks can be found based on projected container flows between seaports and inland terminals. Developing this tool is essential to the project's success, as it will be used to evaluate different scenarios and find the best THB network to be implemented in the pilot project. The higher the quality of the THB networks found by the tool, the more competitive the transport services in the network will be, increasing the chances of the pilot becoming a success.

Therefore, an algorithm that finds good-quality THB networks had to be designed and incorporated in the tool. In order to create this algorithm, four research questions were formulated, which will be answered below.

1. **What existing research has been done on problems similar to the THB problem?**
   In Chapter 2, a survey of the existing literature is presented. While there exists a large body of research on transport planning and scheduling problems, little research has been done to model or solve intermodal freight transport problems similar to the THB problem. Important concepts that have been introduced in the THB problem that have not been modelled in existing intermodal transport literature are the simultaneous consideration of both road- and rail transport alternatives, the organisation of trains in 'batches' that meet at a central hub simultaneously to exchange containers, and the modelling of so-called indirect trains, which start and end at a seaport terminal and an inland terminal while visiting the hub in a batch, enabling some containers on the train to be exchanged with

other trains, while other containers do not partake in the sorting process, saving costs.

2. **Can we characterise the complexity of the THB problem?**
   Investigating whether the THB problem can be solved efficiently is required to determine what kind of algorithms can be used to solve the THB problem. Therefore, a restriction of the original problem definition is given in Chapter 3, which is then used to prove the decision variant of this restriction to be NP-Complete. This means that the original THB problem is an NP-hard optimisation problem, and that any algorithm able to solve this problem to optimality will use an exponential amount of time, unless $P = NP$.

3. **If similar problems exist in literature, can we apply the algorithms that solve this problem to solve the THB problem? If not, can we design an algorithm that solves this problem?**
   Even though we have seen in the literature survey that the THB problem has some unique features that have not been modelled or solved in existing literature, some parts of the modelling of other problems can still be used to create an algorithm for the THB problem. We have proven the THB-problem to be NP-hard, warranting the use of exponential-time algorithms. However, these algorithms need to be anytime algorithms (i.e. algorithms that can be stopped at any moment and return the best solution found so far) to meet the performance requirements of the bundling tool. Chapter 4 presents an Integer Linear Programming (ILP) algorithm for the THB-problem, based on the ILP formulation of an intermodal transport optimisation problem described in a paper by Newman and Yano [29]. A relaxation of the ILP formulation was developed, too. While solutions to this relaxation are not valid solutions to the THB problem, the best solution found can be used as a start solution for the ILP algorithm, which enables the ILP algorithm to find good-quality solutions faster.

4. **Can we characterise the performance of the algorithm(s)?**
   To assure that the algorithms meet the performance requirements, an experimental evaluation of these algorithms is required. In Chapter 5, the performance of the algorithms is evaluated by running them both on a real-world problem, as well as on problems with similar characteristics, but produced by a generator that has been created for this purpose. The generator allows us to create multiple problem instances of different sizes, which can then be used to evaluate the problem's properties. The evaluation has shown that the ILP algorithm produces good-quality solutions to the real-world instance within the required time limit of 24 hours. Tests show that, when running the algorithm for just 30 minutes, finding the optimal solution is only possible for instances larger than 8

source terminals and 8 destination terminals, with at most 4 batches. It is not expected that significantly larger instances can be solved within 24 hours. Using the relaxation in combination with the ILP-algorithm has yielded slightly better results in some cases, but slightly worse results in other cases. The difference in solution quality between running the ILP-algorithm with and without the solution created using the relaxation was not significant for the smaller testcases. Nevertheless, the relaxation does prove its usefulness when running large problems for a relatively small time, with better results for the real-world problems being found much faster than with the ILP-algorithm alone. The results show that both the ILP-algorithm and the combination using the relaxation and the ILP-algorithm (cascade algorithm) meet the performance requirements, meaning that both algorithms are included in the bundling tool.

The most important requirements for the bundling tool – the performance requirements – have been met by the developed algorithms. The other requirements of the bundling tool specified the capabilities of the bundling tool in terms of control requirements that specify the parameters that should be controllable when running the algorithm, and the user interface requirements that specify the actions that should be supported by tool.

The bundling tool provides a comprehensive interface to use the algorithms, allowing for easy evaluation of different problem instances and the effect of different parameter settings for these instances, meeting the control requirements. After the algorithm has finished, the tool shows the best result found by the algorithm, and allows the user to view and compare this solution with intermediate solutions found in the same run, or solutions found using different parameter settings. Statistics are presented with each solution to compare two solutions more quickly, and the detailed view of each batch aids in selecting promising Twin Hub Networks. The user interface therefore meets and exceeds the set requirements by presenting a flexible interface for specifying a problem, running the algorithm, and evaluating the results.

In conclusion, the tool meets all requirements, and can be used in the Twin Hub project to find promising Twin Hub Bundling networks, which can then be put into practice in the upcoming pilot.

7.2 Contributions

Even though the goal of this thesis project was to create a bundling tool, this thesis project has also made a number of contributions to the field of intermodal transport research in the process of developing the tool. The four most important contributions are highlighted below:

1. **Extended existing intermodal transport optimisation problem models** Existing models of intermodal transport optimisation problems do not

include the explicit scheduling of all intermodal transport options at the same time. By allowing all types of transportation to be available to all containers, a schedule can be created that optimises the costs per container over all transport methods, instead of focussing on one transport option (e.g. trains) and allocating the containers not transported by the trains to trucks. A second innovation is the introduction of the concept of batches. Creating these batches not only simplifies the search for THB-networks, as these are groups of indirect trains, but also eliminates the need of including time as a factor in the modelling. Instead, the (discretised) time is replaced by an ordinal system of batches, in which containers put on the stack in one batch can only be retrieved in later batches. The modelling in batches has the distinct advantage that it reduces the size of the model – one batch would typically be modelled as multiple time units. The more options included for the trains in the model, the harder it is to find a solution. This effect was demonstrated in Section 5.2.6. Creating a schedule that could be used in practice would require a large number of time units, making the problem infeasible.Through the organisation in batches, we make explicit that a scheduling step is required after the THB-problem has been solved.

2. **Proved the THB problem to be an NP-hard optimisation problem** Due to the innovative features included in the modelling of the THB-problem, the model has become significantly different from any other optimisation problem in the intermodal transport literature. To confirm that the THB-problem is an NP-hard optimisation problem, we have proved the decision variant of the THB-problem (THB-dec) to be NP-Complete by providing a reduction from the PARTITION-problem to a restriction of the THB-dec problem. As the THB-dec problem is NP-Complete, the THB-problem is an NP-hard optimisation problem.

3. **Created two algorithms for solving the THB-problem** Two algorithms were created to solve the THB-problem; both are based on Integer Linear Programming (ILP). The first algorithm is an exact algorithm incorporating all aspects of the THB problem. As running the algorithm until optimality is not feasible – the THB-problem is an NP-hard optimisation problem after all – the algorithm can be stopped at any time to return the best solution found so far (i.e. it is an anytime algorithm). The ILP-algorithm creates good-quality solutions to real-world problems within a reasonable time frame (24 hours). The second algorithm is based on a relaxation of the ILP-formulation and is able to find solutions much faster. The modelling of indirect trains is omitted, causing a reduction in model size of over 75% for real-world sized models. The solutions found using the relaxation can then be used as an initial solution to start the ILP-algorithm with, speeding up the search for good-quality solutions.

4. **Analysed the performance of the two algorithms** Both the ILP-algorithm and the combination of using the relaxation and the ILP-algorithm ('cascade algorithm') are, in theory, capable of finding the optimal solution for the THB-problem. Experimental analysis has shown that for real-world sized instances, good-quality solutions can be found in a reasonable time frame. While the ILP-algorithm cannot find the optimal solution for real-world instances in reasonable time, it can find the optimal solution for very small problem instances, such as a problem with 8 source terminals, 8 destination terminals, and 4 batches. When running the cascade algorithm on the real-world instance, it is significantly faster in the first part of the execution. However, at some moment, the cascade algorithm performs worse, after which it performs better again. The differences are small, but this effect was unexpected. The hypothesis that the optimal solution of the relaxation is in a local optimum for the ILP-algorithm was rejected, so no explanation for this phenomenon can be given. For smaller instances, however, the cascade algorithm does not perform significantly better. Two questions about the properties of the algorithms were also evaluated. We have shown that the number of batches should be chosen carefully, as when a maximum is chosen that is too high, the performance is degraded significantly. Furthermore, we have shown that the asymmetry in the problem formulation, where the number of variables required to model indirect trains is linear in the number of sources, but quadratic in the number of destinations, also has a significant effect on the quality of the solution, meaning that it is better to mirror the instance when the number of destinations is larger than the number of sources.

7.3 Future Work

Even though all requirements set for this thesis project have been met, there are many interesting research questions which unfortunately have to be left unanswered in this thesis due to limitations in time. Four suggestions for future work are presented below.

- **Extend the model to include full round-trips:** The current modelling of the THB problem only bundles the flows in one direction, i.e. it bundles the flows from the seaports to the inland terminals or vice versa. This may cause imbalances in the produced schedules, as the flows in one of the two directions may not be sufficient to create a profitable train service. This would mean that trains have to be moved to another location, incurring additional costs which are not modelled in the current formulation. This is a known problem in transportation research, and it has been identified in the taxonomy by Crainic and Laporte [9] as 'crew and motive power scheduling'. By modelling both directions at the same time, bundling

networks can be identified where it is profitable to create a service in both directions, eliminating these problems.

- **Investigating the possibilities for developing an approximation algorithm:** For many NP-hard optimisation problems, it is possible to develop an approximation algorithm that delivers a guaranteed performance in terms of objective value, while taking only a polynomial amount of time. Investigating whether an approximation algorithm can be created for the THB problem would not only be interesting from a research perspective, but could also perform better than the ILP-algorithm in terms of objective value found after a certain amount of computation time when an (F)PTAS could be created.

- **Creating a polynomial-time heuristic** The cascade algorithm performs better than the ILP-algorithm on the real-world instance in the first hours of computation, due to the better initial solution found by solving the relaxation. However, this solution does not contain indirect trains, while these are used almost exclusively in the optimal solution of the THB-problem. Creating a heuristic that is able to find good-quality solutions containing indirect trains in polynomial time could, therefore, cause a much larger speed-up when running it in combination with the ILP-algorithm, enabling the ILP-algorithm to find better solutions faster.

- **More detailed evaluation of the performance of the cascade algorithm on real-world problems:** The cascade algorithm generally performs better, and has a large advantage over the ILP-algorithm in the first part of the algorithm's execution. However, this advantage gradually disappears at some point during the execution, where the ILP-algorithm performs better. Later on in the execution process, the cascade algorithm performs better once more. Investigating the causes of this behaviour may lead to new insights for finding a more efficient relaxation.

Additionally, the tool can be extended so the mirroring of instances that have more destination terminals than source terminals happens automatically, as we have already proven that this yields better results faster.

Bibliography

[1] Y.M. Bontekoning. *Hub exchange operations in intermodal hub-and-spoke operations: comparison of the performances of four types of rail-rail exchange facilities*. PhD thesis, OTB Research Institute, 2006.

[2] YM Bontekoning and E. Kreutzberger. *Concepts of new-generation terminals and terminal-nodes*. Delft Univ. Press, 1999.

[3] P. Cardebring, R. Fiedler, C. Reynaud, and P. Weaver. Analysing intermodal quality: a key step toward enhancing intermodal performance and market share in europe. *Summary Report of the IQ Project (listed within the IVth Framework Program of EC) disponible à: http://www. tfk-hamburg. com*, 2000.

[4] A. Caris, C. Macharis, and G.K. Janssens. Planning problems in intermodal freight transport: accomplishments and prospects. *Transportation Planning and Technology*, 31(3):277–302, 2008.

[5] CEI-Demeyer NV. Main hub terminal antwerp north. `http://www.cei-demeyer.be/files/wharves/en/burgerlijke_bouwkunde/spoorwerken/terminal_main_hub.pdf`, 1999.

[6] European Commission. Freight transport logistics in europe–the key to sustainable mobility, 2006.

[7] J.F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32(4):380–404, 1998.

[8] T. Crainic, J.A. Ferland, and J.M. Rousseau. A tactical planning model for rail freight transportation. *Transportation Science*, 18(2):165–184, 1984.

[9] T.G. Crainic and G. Laporte. Planning models for freight transportation. *European Journal of Operational Research*, 97(3):409–438, 1997.

[10] T.G. Crainic and J.M. Rousseau. Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Research Part B: Methodological*, 20(3):225–242, 1986.

[11] G.B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. *New York*, 1951.

[12] Dutch Government. Nederland belangrijke handelspartner voor alle eu-landen. http://www.rijksoverheid.nl/nieuws/2012/08/02/nederland-belangrijke-handelspartner-voor-alle-eu-landen.html, August 2012. (in Dutch) Retrieved 22-08-2012.

[13] European Commission. Intermodal transport: The marco polo programme. http://europa.eu/legislation_summaries/transport/intermodality_transeuropean_networks/l24159_en.htm, July 2003.

[14] B. Groothedde. *Collaborative logistics and transportation networks: a modelling approach to hub network design*. PhD thesis, TRAIL Research School, 2005.

[15] T. Grünert and H.J. Sebastian. Planning models for long-haul operations of postal and express shipment companies. *European Journal of Operational Research*, 122(2):289–309, 2000.

[16] Gurobi Optimization. Gurobi. http://www.gurobi.com/, .

[17] Gurobi Optimization. Gurobi reference manual: Parameters. http://www.gurobi.com/documentation/5.0/reference-manual/node653, . Retrieved 25-08-2012.

[18] A.E. Haghani. Formulation and solution of a combined train routing and makeup, and empty car distribution model. *Transportation Research Part B: Methodological*, 23(6):433–452, 1989.

[19] IBM. Cplex optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[20] M. Janic, A. Reggiani, and T. Spicciarelli. The european freight transport system: theoretical background of the new generation of bundling networks. In *World Transport Research: Selected Proceedings of the 8th World Conference on Transport Research*, number Volume 1, 1999.

[21] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[22] Richard M Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 40(4):85–103, 1972.

[23] E. Kreutzberger. The innovation of intermodal rail freight bundling networks in europe. *Concepts, Developments, Performances, TRAIL Thesis Series*, (T2008/16), 2008.

[24] E. Kreutzberger. From bundling theory to network and node innovation: Intermodal rail freight twin hub network antwerp/rotterdam (irthnar). 2011.

[25] E.D. Kreutzberger. Project proposal: Intermodal rail freight twin hub network northwest europe. Version May 2011.

[26] E.D. Kreutzberger. Distance and time in intermodal goods transport networks in europe: A generic approach. *Transportation Research Part A: Policy and Practice*, 42(7):973–993, 2008.

[27] E.D. Kreutzberger, R. Konings, and C. Witteveen. Modelling the bundling of intermodal rail flows from/to seaports, 2008.

[28] C. Macharis and YM Bontekoning. Opportunities for or in intermodal freight transport research: A review. *European Journal of Operational Research*, 153(2):400–416, 2004.

[29] A.M. Newman and C.A. Yano. Centralized and decentralized train scheduling for intermodal operations. *IIE Transactions*, 32(8):743–754, 2000.

[30] A.M. Newman and C.A. Yano. Scheduling direct and indirect trains and containers in an intermodal setting. *Transportation science*, 34(3):256–270, 2000.

[31] L.K. Nozick and E.K. Morlok. A model for medium-term operations planning in an intermodal rail-truck service. *Transportation research part a: policy and practice*, 31(2):91–107, 1997.

[32] Port of Rotterdam. Totale goederenoverslag 2010. `http://www.portofrotterdam.com/nl/Over-de-haven/havenstatistieken/Documents/totale_goederenoverslag_2010.pdf`, 2011.

[33] H. Rotter. New operating concepts for intermodal transport: The mega hub in hanover/lehrte in germany. *Transportation Planning and Technology*, 27(5):347–365, 2004.

[34] Statistics Netherlands. Statline statistics 2010. `http://statline.cbs.nl/StatWeb/?LA=en`, 2010. Retrieved 22-07-2012.

[35] Statistics Netherlands. Belangrijkste handelspartners van eu-landen en het aandeel van nederland in de totale handel van eu-landen. http://www.cbs.nl/nl-NL/menu/themas/internationale-handel/links/2012-3661-mw.htm, 2011. (in Dutch) Retrieved 22-08-2012.

[36] Statistics Netherlands. Internationalisation monitor 2011. http://www.cbs.nl/nl-NL/menu/themas/internationale-handel/publicaties/publicaties/archief/2011/2011-m21-2-eng-pub.htm, October 2011. Retrieved 22-08-2012.

Appendix A

Complete definition of the THB Problem

Below, the complete definition of the Twin Hub Bundling (THB) problem is presented.

TWIN-HUB BUNDLING (THB):
Given:

- A set of inland terminal nodes I with $|I| \in \mathbb{N}^*$
- A set of port terminal nodes P with $|P| \in \mathbb{N}^*$
- The hub node h
- $V = \{h\} \cup I \cup P$
- $E^{direct} \subseteq (I \times P) \cup (P \times I)$
- $E^{hub} \subseteq (I \times \{h\}) \cup (\{h\} \times I) \cup (P \times \{h\}) \cup (\{h\} \times P)$
- $E = E^{direct} \cup E^{hub}$
- A list of possible batches B with $|B| \in \mathbb{N}^*$
- The desired frequency of train services per week S.
- The flow of containers F_{ij} between terminals i and j with either $i \in I, j \in P$ or $i \in P, j \in I$.
- The maximum number of trains that can exchange load units at the hub at the same time H, i.e. the capacity of one batch $b \in B$.
- The half economical round-trip time Q, which is equal to an integer multiple of 12 hours.
- The fixed cost per hour for using a train C_{time}^{fixed}. Note that use is defined as being part of the economical round-trip time of the train, not the time the train is actually driving.

- The fixed cost per kilometre for driving a train C_{km}^{fixed}.
- The variable cost per load unit per hour for using a train $C_{time}^{variable}$.
- The variable cost per load unit per kilometre for driving a train $C_{km}^{variable}$.
- The total cost per hour for using a truck C_{time}^{truck}.
- The total cost per kilometre for using a truck C_{km}^{truck}.
- The hub transfer costs of one load unit $C^{transfer}$.
- The costs for pre- or post-haulage for one load unit C^{pph}.
- For each edge $e_i \in E$ an associated distance $dist(e)$ in kilometres.
- Distance brackets denoting the average speed of a train in kilometres per hour related to the distance it travels: $(D_i^{min}, D_i^{max}) \rightarrow U_{avg}^{train}$.
- The average speed U_{avg}^{truck} of a truck.
- The maximum capacity W of a train.
- The maximum stack size M.

How can each flow F_{ij} be allocated to a set of trains T and trucks R that may participate in one of the batches in B, while minimizing the following cost function:

$$\begin{aligned} Z &= \sum_{t \in T} (cost^{fixed}(t) + cost^{variable\_direct}(t) + cost^{variable\_hub}(t) + cost^{variable\_dest}(t)) \\ &+ \sum_{r \in R} (cost^{total}(r)) \end{aligned}$$

Where:

- v_t^{start} = the start terminal of transport t
- v_t^{end} = the end terminal of transport t
- $dist(t) = \sum_{\{e \in E \ | \text{train travels on e}\}} e$, dist(r) is defined similarly, but for trucks
- $time(t) = \lceil \frac{dist(t)}{V_{avg}^{train} \times Q} \rceil \times Q$ with V_{avg}^{train} being the average speed for distance dist(t).
- $time(r) = \lceil \frac{dist(t)}{V_{avg}^{truck}} \rceil$
- $L_{v_1,v_2,t}$ = load transported from v_1 to v_2 on train t with $v_1, v_2 \in V \setminus \{h\}$.

- $L_{v_1,h,t} = \sum_{v_2 \in V \setminus \{h\}, v_1} L_{v_1,v_2,t}$
- $L_{h,v_2,t} = \sum_{v_1 \in V \setminus \{h\}, v_2} L_{v_1,v_2,t}$
- $cost^{fixed}(t) = C^{fixed}_{time} * time(t) + C^{fixed}_{km} * dist(t)$
- $cost^{variable\_direct}(t) = L_{v_t^{start},v_t^{end},t}(C^{variable}_{time} * time(t) + C^{variable}_{km} * dist(t) * +2 * C^{pph})$
- $cost^{variable\_hub}(t) = L_{v_t^{start},h,t}(C^{variable}_{time} * time(e_{v_t^{start},h}) + C^{variable}_{km} * dist(e_{v_t^{start},h}) + C^{pph} + 0.5 * C^{transfer})$
- $cost^{variable\_dest}(t) = L_{h,v_t^{end},t}(C^{variable}_{time} * time(e_{h,v_t^{end}}) + C^{variable}_{km} * dist(e_{h,v_t^{end}}) + C^{pph} + 0.5 * C^{transfer}))$
- $cost^{total}(r) = C^{truck}_{time} * time(r) + C^{truck}_{km} * dist(r) + 0.5 * C^{transfer}(L_{v_1,h,r} + L_{h,v_2,r})$

Under the following constraints:

1. Trains and trucks can only exchange load units with each other if they belong to the same batch $b \in B$.
2. Batches have an order in which they are processed, allowing for interaction of trains and trucks with the stack so that containers can be temporarily stored between batches b_i and b_j provided that $i \leq j$.
3. The desired frequency S is achieved on all connection-pairs in the system where $F_{ij} > 0$.
4. The resulting schedule is periodical.
5. After each batch, the number of containers stored on the stack is at most M.

Appendix B

Bounds on decision variables of IP algorithm

The bounds placed on the decision variables as stated in Table B.3 have to be as low as possible, as this will reduce the solution space and hence will speed up the algorithm. Most bounds are straight-forward, for example the bounds on the direct trains and the direct trucks are calculated in such a way that all containers from origin to destination can be transported using these direct connections. In a similar way, the bounds on the number of trains from origin to hub and from hub to destination are limited by the amount of trains that can be filled with the combined flow towards the hub or towards the destination, and by the number of trains that each batch may contain.

The amount of containers transported by direct trains is limited to the number of containers being transported from an origin to a destination.

When looking at the operations at the hub, the amount of transfers in a batch is limited by the maximum length of a train at the hub, and by the amount of trains that can visit the hub in a batch. Multiplying these figures gives an upper bound on the amount of transfers.

For the other upper bounds on the decision variables, certain assumptions have to be made. These assumptions are listed in Table B.1.

Lemma B.1. *For any source-destination pair $i \in I,\ j \in J$ in a batch $b \in B$, an optimal solution O exists where $t_{ijb}^{indirect}$ takes only binary values.*

Proof. Suppose an optimal solution O' exists where $t_{ijb}^{indirect}$ takes an integer value $p > 1$. We discern two cases:

1. The number of containers that are not transferred $c_{ijb}^{indirect} \geq (p-1) \times W$: We transform the solution O' into O by adding $p-1$ to t_{ij}^{direct}, and $(p-1) \times W$ to c_{ij}^{direct}. This solution is still optimal, as assumptions B.3 and B.5 state that direct trains have costs smaller than or equal to indirect

Assumptions

$$f_{ij}^{direct} + W \times v_{ij}^{direct} \leq W \times r_{ij}^{direct} \quad \forall (i \in I, j \in J) \tag{B.1}$$

$$r_{ij}^{direct} \leq r_{ijb}^{hub} + r_{ijb}^{dest} \quad \forall (i \in I, j \in J, b \in B) \tag{B.2}$$

$$f_{ij}^{direct} \leq f_{ij}^{indirect} \quad \forall (i \in I, j \in J) \tag{B.3}$$

$$f_{ij}^{indirect} \leq f_{i}^{hub} + f_{j}^{dest} \quad \forall (i \in I, j \in J) \tag{B.4}$$

$$v_{ij}^{direct} \leq v_{ij}^{indirect} \quad \forall (i \in I, j \in J) \tag{B.5}$$

$$v_{ij}^{indirect} \leq v_{i}^{hub} + v_{j}^{dest} \quad \forall (i \in I, j \in J) \tag{B.6}$$

All costs are assumed to be strictly greater than 0 (B.7)

Table B.1: Assumptions made for calculating the upper bounds for the decision variables

trains. We have now obtained an optimal solution O conforming to the lemma.

2. The number of containers that are not transferred $c_{ijb}^{indirect} < (p-1) \times W$: There are two sub-cases here:

   a) The total number of containers on the trains $\sum_{j}(c_{ijb}^{indirect} + \sum_{j' \in J, j \neq j'} c_{ijj'b}^{indirect\_transfer}) \leq (p-1) \times W$: We transform the solution O' by changing $t_{ijb}^{indirect}$ to $p-1$. This solution contains less trains, and as we assume nonnegative costs (assumption B.7), cheaper than O'. This is in contradiction with the optimality of O'.

   b) The total number of containers on the trains $\sum_{j}(c_{ijb}^{indirect} + \sum_{j' \in J, j \neq j'} c_{ijj'b}^{indirect\_transfer}) > (p-1) \times W$: We must now have at least

one $j' \in J$ for which the flow in the batch b is greater than 0. Let us call the original amount of indirect trains from i to j in batch b ($z^{indirect}_{ij'b}$) 'q'. We know that the flow $c^{indirect}_{ijb}$ is smaller than $(p-1) \times W$. Consider now the valid solution $\mathcal{S}$ with $t^{indirect}_{ijb} = p-1$ and $z^{indirect}_{ij'b} = q+1$. Now we allocate the containers that were first transferred on another train $c^{indirect\_transfer}_{ijj'b}$ and allocate these to $x^{indirect}_{ij'b}$. The solution $\mathcal{S}$ now has less transfers, and is, as we assume non-negative transfer costs (assumption B.7), cheaper than O'. This is in contradiction with the optimality of O'.

We have shown that an optimal solution O' can be transformed into the optimal solution O in which $t^{indirect}_{ijb}$ takes only binary values. □

For the calculation of the bound of the number of containers being transported by indirect trains, assumptions B.3 and B.5 state that it is at least as cheap to let a direct train drive than it is to let an indirect train drive the same route. These assumptions are valid: the distance an indirect train travels can only be greater than or equal to the point-to-point distance, as the direct train can take a route past the hub and not attend the hub, and travel the same distance. The decision variable for the indirect trains can, due to this assumption and Lemma B.1, be made binary. Consequently, the number of containers transported on an indirect train in one batch is also bounded by the number of containers that can be transported on one train. This also holds for the amount of containers that require a transfer at the hub.

Lemma B.2. *For any source-destination pair $i \in I$, $j \in J$ in a batch $b \in B$, the number of containers transported on hub trains c^{hub}_{ijb} is bounded above by W in any optimal solution O.*

Proof. Suppose more than W containers are transported from i to j in a single batch on hub trains. Now consider an alternative solution $\mathcal{S}$ where n direct trains from i to j are added with $n \times W \leq c^{hub}_{ijb}$. Remove the containers from the transport methods that transported the containers to their final destination, and account for the reduction in transfers. As full direct trains cost the same as or are cheaper than other forms of transport, and the amount of transfers is now reduced, solution $\mathcal{S}$ is cheaper than O. This is a contradiction with the assumption that O is optimal. Therefore, the decision variable c^{hub}_{ijb} is bounded above by W. □

Lemma B.3. *For any source-destination pair $i \in I$, $j \in J$ in a batch $b \in B$, the number of containers transported on destination trains c^{dest}_{ijb} is bounded above by W in any optimal solution O.*

Proof. Suppose more than W containers are transported from i to j in a single batch on destination trains. Now consider an alternative solution $\mathcal{S}$ where n

direct trains from i to j are added with $n \times W \leq c_{ijb}^{dest}$. Remove the containers from the transport methods that transported the containers to the hub, and account for the reduction in transfers. As full direct trains cost the same as or are cheaper than other forms of transport, and the amount of transfers is now reduced, solution $\mathcal{S}$ is cheaper than O. This is a contradiction with the assumption that O is optimal. Therefore, the decision variable c_{ijb}^{dest} is bounded above by W. □

The combination of a hub transport and a destination transport – train or truck – can never be cheaper than letting an indirect train drive (assumptions B.4 and B.6), as the two trains drive the same route as the indirect train. While the upper bound on the amount of hub trains and destination trains is bounded by the possible container flow and the number of trains in a batch, the amount of containers that can be transported by such trains is limited to the capacity of one full train, as is shown in Lemmas B.2 and B.3.

Assumption B.2 states that it is at least as cheap to let a direct truck drive than to let one hub truck and one destination truck drive, as the truck will never drive a cheaper route when driving through or even stopping at the hub.

Finally, the inventory at the hub is bounded as stated in Lemma B.4.

Lemma B.4. *For any source $i \in I$, destination $j \in J$, in a batch $b \in B$, there exists an optimal solution O where the inventory size A_{ijb}^{hub} is bounded by* $\min(W, F_{ij})$.

Proof. There can never be more containers present at the hub than there are in the system. It is therefore obvious that the variable is bounded by F_{ij}. Furthermore, the amount of containers stored at the hub with origin i and destination j can never exceed W. Otherwise, we can follow the reasoning of Lemma B.3:

Consider an alternative solution $\mathcal{S}$ where n direct trains from i to j are added with $n \times W \leq c_{ijb}^{dest}$. Remove the containers from the transport methods that transported the containers to the hub and to the destination, and account for the reduction in transfers. As full direct trains cost the same as or are cheaper than other forms of transport, and the amount of transfers is now reduced, solution $\mathcal{S}$ is cheaper than O. This is a contradiction with the assumption that O is optimal. Therefore, the inventory at the hub A_{ijb}^{hub} is also bounded above by W. □

Parameter		**Description**
I	=	set of origins
J	=	set of destinations
B	=	list of batches
f_{ij}^{direct}	=	fixed cost of running a train directly between $i \in I$ and $j \in J$
$f_{ij}^{indirect}$	=	fixed cost of running a train between $i \in I$ and $j \in J$ while visiting the hub
f_i^{hub}	=	fixed cost of running a train between $i \in I$ and the hub
f_j^{dest}	=	fixed cost of running a train between the hub and $j \in J$
v_{ij}^{direct}	=	variable unit cost of transporting a container directly from $i \in I$ to $j \in J$, including pre- and post-haulage
$v_{ij}^{indirect}$	=	variable unit cost of transporting a container from $i \in I$ to $j \in J$ while visiting the hub, including pre- and post-haulage
v_i^{hub}	=	variable unit cost of transporting a container from $i \in I$ to the hub, including pre-haulage
v_j^{dest}	=	variable unit cost of transport a container from the hub to $j \in J$, including post-haulage
R_{ij}^{direct}	=	unit cost of transporting a container directly from $i \in I$ to $j \in J$ by truck
R_i^{hub}	=	unit cost of transporting a container from $i \in I$ to the hub by truck
R_j^{dest}	=	unit cost of transporting a container from the hub to $j \in J$ by truck
$C^{transfer}$	=	cost of transferring a container from one transport to another at the hub
W	=	maximum capacity of a train
H	=	maximum number of trains that can attend the hub simultaneously in a batch
M	=	maximum number of containers that can be stored on the stack
F_{ij}	=	number of containers to be transported from $i \in I$ to $j \in J$

Table B.2: Parameters of the ILP problem

Decision variable		**Description**	**Bounds**
t_{ij}^{direct}	=	number of trains driving directly between $i \in I$ and $j \in J$	$[0, \lceil F_{ij}/W \rceil]$
$t_{ijb}^{indirect}$	=	number of trains driving between $i \in I$ and $j \in J$ while visiting the hub belonging to batch $b \in B$	$[0,1]$
t_{ib}^{hub}	=	number of trains driving between $i \in I$ and the hub belonging to batch $b \in B$	$[0, \min(\sum_{j \in J} \lceil F_{ij}/W \rceil, H)]$
t_{jb}^{dest}	=	number of trains driving between the hub and $j \in J$ belonging to batch $b \in B$	$[0, \min(\sum_{i \in I} \lceil F_{ij}/W \rceil, H)]$
c_{ij}^{direct}	=	number of containers transported from $i \in I$ to $j \in J$ by a direct train	$[0, F_{ij}]$
$c_{ijb}^{indirect}$	=	number of containers transported from $i \in I$ to $j \in J$ by an indirect train visiting the hub in batch $b \in B$ where no transfer is required	$[0, \min(F_{ij}, W)]$
$c_{ijj'b}^{indirect\_transfer}$	=	number of containers transported from $i \in I$ to $j' \in J$, being transported on an indirect train headed for destination $j \in J$, and visiting the hub in $b \in B$	$[0, \min(F_{ij}, W)]$
$c_{ijb}^{indirect\_hub}$	=	number of containers transported from $i \in I$ to the hub destined for $j \in J$, belonging to batch $b \in B$, on an indirect train	$[0, \min(F_{ij}, W)]$
$c_{ijb}^{indirect\_dest}$	=	number of containers transported from the hub to $j \in J$, having origin $i \in I$ and belonging to batch $b \in B$, on an indirect train	$[0, \min(F_{ij}, W)]$
c_{ijb}^{hub}	=	number of containers transported from $i \in I$ to the hub destined for $j \in J$, belonging to batch $b \in B$, on a train ending at the hub	$[0, \min(F_{ij}, W)]$
c_{ijb}^{dest}	=	number of containers transported from the hub to $j \in J$, having origin $i \in I$ and belonging to batch $b \in B$, on a train starting at the hub	$[0, \min(F_{ij}, W)]$
r_{ij}^{direct}	=	number of containers transported directly from $i \in I$ to $j \in J$ by truck	$[0, F_{ij}]$
r_{ijb}^{hub}	=	number of containers transported from $i \in I$ to the hub by truck, destined for $j \in J$, belonging to batch $b \in B$	$[0, F_{ij}]$
r_{ijb}^{dest}	=	number of containers transported from the hub to $j \in J$ by truck, having origin $i \in I$ and belonging to batch $b \in B$	$[0, F_{ij}]$
$c_{b}^{transfer}$	=	number of containers transferred from one transport (train or truck) to another at the hub during batch $b \in B$	$[0, H \times W]$
A_{ijb}^{hub}	=	number of containers that are stored after batch $b \in B$ at the hub with origin $i \in I$ and destination $j \in J$	$[0, \min(W, F_{ij})]$

Table B.3: Decision variables for the ILP problem

Appendix C

Bundling Tool Documentation

No software project is complete without some form of documentation and design rationale, and the Twin Hub Bundling Tool is no exception. In this chapter, a requirements analysis is presented, as well as documentation concerning the structure of the bundling tool implementation, and the specification of the file structures used by the tool.

C.1 Requirements analysis

The functionality of the Twin-Hub bundling tool consists of three features. First, the user must be able to specify the problem instance to be solved by the algorithm and the parameters with which to call the algorithm. Second, the tool must call the algorithm with the required parameters and report the progress of the algorithm back to the user. Third, the tool must present the results of the algorithm – the bundling networks that have been generated – to the user together with relevant statistics that help interpret this result.

Each of these features will be developed incrementally in collaboration with the customer.

The requirements analysis consists of a set of functional and non-functional requirements, and a set of constraints on the program. The functional requirements are specified in Section 1.3, and the non-functional requirements and constraints are specified below. Then, a system model of the tool is presented.

C.1.1 Non-functional requirements

The non-functional requirements are those requirements that do not directly involve one specific goal of the user. However, these requirements do have an impact on the overall user experience, and are necessary to help the user achieve its goals.

1. **User interface and human factors**
   The users of this tool are not assumed to be experts in computer science, and may have limited experience with using command-line tools. The interface

of the tool must allow for entering the large number of parameters in an easy way, and must present the results, together with the calculated statistics, in an intelligible manner. There are no special requirements concerning human factors.

2. **Hardware considerations**
   The tool will require as much processing power as possible, as the algorithms included in the tool are very computationally intensive. The Integer Linear Programming algorithm and the algorithm using the relaxation can fully utilize all cores in one machine in an effort to solve the problem. The recommended amount of system memory available for the algorithm is 2 gigabytes per core to solve most real-world sized problems (i.e. around 25 port terminals and 40 inland terminals). For problem sizes that are significantly larger, more memory is required for the algorithm to function properly. The free disk space required by the algorithm for a typical real-world sized problem is 20 gigabytes, depending on the amount of system memory installed – systems with more memory may require less free disk space.

3. **Performance characteristics**
   It is important that the user receives regular feedback on the progress of the algorithm, and that the optimization process can be aborted at all times during the execution of the algorithm. Stopping the algorithm may take at most five seconds when no results have to be saved, and at most one minute when the solutions have to be saved. The user interface must remain responsive at all times, even when the algorithm is running.

4. **Error handling and extreme conditions**
   The tool must give adequate feedback on errors that may occur during execution, such as parameters that have been entered incorrectly, or errors that might occur during the execution of the algorithm.

   The tool will deliver best-effort performance when system memory is limited, but performance may degrade significantly. When the available disk space is low, the algorithm may have to stop during its execution and may not be able to return a result.

5. **System interfacing**
   The tool will not interface with other programs, and will only interact with the machine it runs on.

6. **Quality issues**
   The tool does not have special quality issues.

7. **System modifications**
   The proposed system will completely replace the functionality of the existing system, which relied on manual selection of promising Twin-Hub Networks

based on intuition and manual calculations. The new system creates Twin-Hub networks based on optimization criteria instead.

8. **Physical environment**
   No special requirements.

9. **Security issues**
   There are no special security issues when running this tool.

C.1.2 Constraints

The tool has to be built in Java to properly interface with the algorithms. No other constraints are placed on the development of the tool.

C.1.3 System models

The following use cases describe the four main ways in which the user will use the tool.

1. **Setting up the tool**
   Actors: Users wanting to use the tool
   Goal: Configuring the tool so that it can be used to run problem instances
   Preconditions: The user has a working installation of Gurobi Solver (version 4.6.1 or 5.0.x)
   Summary: The user configures the tool to use the Gurobi Solver
   Related use cases: None
   Steps:

Actor actions	System response
Enters the path to the Gurobi installation folder in the configuration file	-
(Re)starts the tool	Shows start screen

   Postcondition: The tool is started and ready to be used

2. **Running the algorithm on a problem instance**
   Actors: Users wanting to evaluate a problem instance
   Goal: Obtaining a Twin-Hub Bundling Network transporting all containers in the given problem instance
   Preconditions: The tool has been correctly set up (1)
   Summary: The problem instance is specified in the tool and the algorithm is called for that instance, the result is then displayed
   Related use cases: Setting up the tool(1)
   Viewing a result (3)
   Viewing multiple results (4)

Steps:

Actor actions	System response
Selects the option to create a new run	Displays fields for specifying the problem instance
Enters required information in fields and clicks the 'run' button	Opens a screen showing the progress of the algorithm.
Either the user waits for the algorithm to complete (2.1), or chooses to abort the algorithm (2.2)	

2.1. **User waits for the algorithm to complete**
Steps:

Actor actions	System response
Waits for algorithm to complete	Shows that the algorithm is done
Clicks 'View result' button	Shows the best result of the run in a new screen

Postcondition: The solution of the problem instance has been computed and is shown in the tool

2.2. **User does not wait for the algorithm to complete**
Steps:

Actor actions	System response
Clicks 'Abort'	Notifies the user that the algorithm has been aborted and progress has been saved. If a result has been found, the tool displays the best result that has been found so far. If not, the tool remains in the progress screen, which can be discarded.

Postcondition: The tool has saved the progress and displays the best result so far, if any

3. **Viewing a result**
Actors: Users wanting to evaluate a result
Goal: Opening a solution file so that it shown in the solution viewer
Preconditions: Setting up the tool (1), Running the algorithm on a problem instance (2)
Summary: The user opens an existing solution
Related use cases: Viewing multiple results (4)
Steps:

Actor actions	System response
Clicks the 'Open solution file' button	Displays a file dialogue for selecting a solution file in the file system
Navigates to folder, selects file, and clicks 'OK'	Shows screen with the solution from the file

Postcondition: The solution has been read from the file, and is shown on the screen

4. **Viewing multiple results**
Actors: Users wanting to compare multiple results
Goal: Configuring the tool so that it can be used to run problem instances
Preconditions: Viewing a result (3)
Summary: The user opens multiple existing solutions and views them simultaneously
Related use cases: Viewing a result (3)
Steps:

Actor actions	System response
Repeats use case 'Viewing a result' (3) multiple times	Opens solutions and displays them on the screen
Moves one solution to one side of the screen	Divides the screen in two parts, with the new part containing the moved solution

Postcondition: The moved solution is displayed in another part of the screen, next to the other solution(s)

C.2 Software architecture of the tool

The architecture of the tool has evolved during each iteration of the development process, and is still prone to change as new features are added or the requirements of existing features are changed. In this section, a high-level overview of the architecture of the tool is presented. Figure C.1 shows a package diagram of the tool. The functionality implemented in each subsystem will be explained below.

C.2.1 Algorithms

The Algorithms subsystem contains the implementation of the algorithms and contains both the Integer Linear Programming algorithm and the cascade algorithm specified in Sections 4.1 and 4.2, as these algorithms are closely related. Both algorithms are based on Integer Linear Programming and interface with the Gurobi solver to solve the problem. All communication with the Gur-

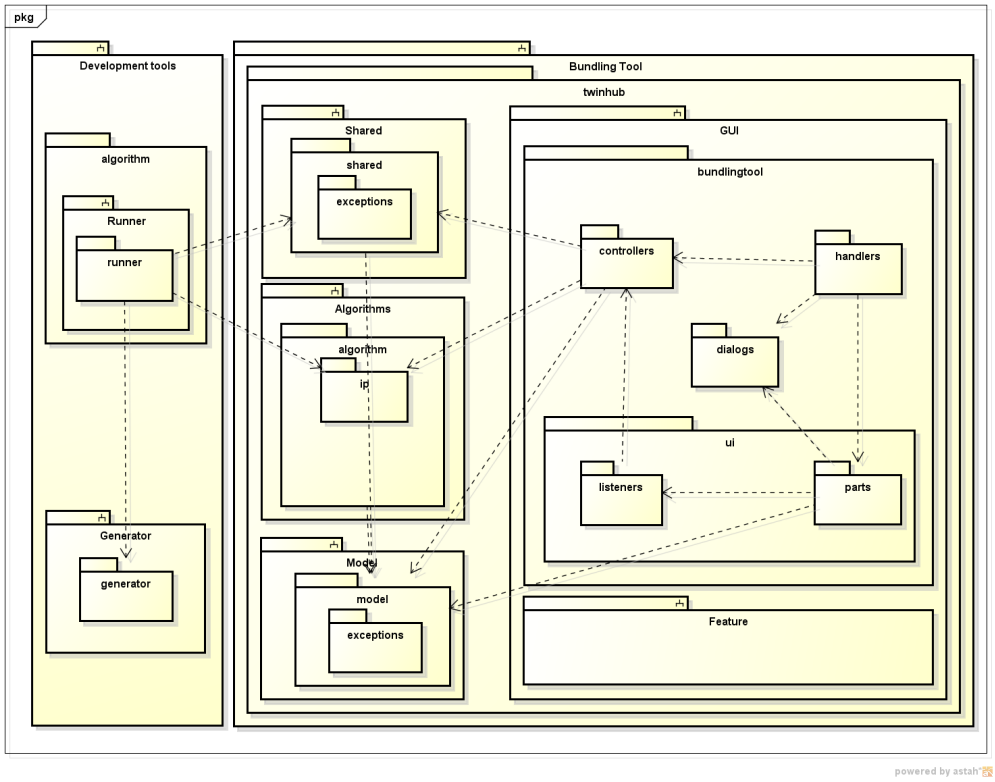

Figure C.1: Package diagram of the Bundling Tool and the Development Tools

obi solver is handled by the classes in this package, including the callback mechanism that is used to communicate the progress of the solver.

All algorithms implement a way of creating the specifications of a run beforehand, and loading this object in the algorithm in order to run that instance. During the execution of the algorithm, the algorithms use the Observer-pattern to pass messages about the progress of the algorithm to the user interface objects. The object calling the algorithm – usually a controller in the GUI – is expected to register itself as an Observer of the algorithm and relay those messages to the GUI-element responsible for reporting progress back to the user.

C.2.2 Model

The Model subsystem contains the model classes of the algorithm and bundling tool. All model classes that are part of a solution file (i.e. all classes except for Train, Truck, and Batch) implement the interface ISavable. This interface indicates that the object is part of the solution file, and requires a method toFileString() to be implemented which returns a string-representation of the object to be saved in the solution file. Another – static – method is then implemented to read this representation and return the original object. Calling the toFileString-method on a solution object will, therefore, return a string

representation of the solution that can be restored later.

The calculation of the costs of a solution is also performed by the model. The Train, Truck, Batch, and Solution classes implement a calculateCosts-method that takes a cost model as an argument and calculates the costs. The Batch and Solution classes calculate their costs based on the Trains and Trucks contained therein.

C.2.3 Shared

The Shared subsystem contains classes for the reading and processing of problem instances and solutions. A solution file in the Gurobi format cannot be interpreted directly by the tool, for example, but it must first be processed to instantiate the correct Train, Truck, Batch, and Solution objects, together with their containers. Other functionality provided by this package include a matrix reader that reads matrices stored in CSV-format, and a problem instance reader.

C.2.4 GUI

The GUI subsystem is the core part of the tool and contains the user interface and the controllers for calling the algorithms and the solution reader. The user interface is created in Eclipse RCP e4, which dictates a certain structure for the GUI. Each subsystem is contained in a plug-in, with explicit dependencies to other plug-ins. A feature project is then created to define all plug-ins that are used by the program.

The GUI is built up of one window containing a tabbed structure. Each tab contains a 'part' specifying the contents of that tab. There are three main parts, one for each feature of the tool: a part to configure a run, a part to show the progress of the algorithm, and a part to view the solutions in. The part for the configuration of a run is also the start screen of the tool; the progress part is only shown when an algorithm is running or has just run, and the solution view part is only opened after the algorithm has finished or a solution has been opened. Opening a solution is done through the open command in the menu or on the toolbar. Each command has a handler associated with it that serves as a listener to the button and calls the appropriate methods in the controller to – in the case of the open command – open a file selector window and create a new tab with the selected solution.

C.2.5 Development Tools

The Development Tools subsystem will not be shipped with the final bundling tool, and is used for testing purposes. It contains a runner for the algorithms that works independently of the GUI and the generator which can be used to generate problem instances. These tools do not have an interface due to the large number of parameters that have to be set, but are controlled through a

class that creates the runs and sets the parameters. All results in the algorithm analysis (Section 5) have been generated using these tools.

C.3 Bundling tool input and output file specifications

The input and output files need to conform to certain specifications in order for them to be used by the bundling tool. The specifications for these files are detailed below.

C.3.1 Flow and distance matrices

The flow and distance matrices are the only input files for the bundling tool that have to be created by hand. Both matrices have to be in CSV-format using semicolons as a separator. These files can be created and edited using any spreadsheet program, such as OpenOffice Calc, Microsoft Excel, or Apple Numbers.

The flow matrix is a $n \times m$ matrix with n the number of source terminals and m the number of destination terminals. Each cell in the matrix contains the flow in number of containers between a source and a destination terminal.

The distance matrix is a $(n+1) \times (m+1)$ matrix with n the number of source terminals and m the number of destination terminals. Each cell in the matrix contains the distance between each source and destination terminal. The last column and last row contain the distances from each source terminal to the hub and from the hub to each destination terminal respectively.

C.3.2 Solutions

The solution file contains the problem instance and parameters with which the algorithm was run, as well as the solution itself. The solution is coded using the Gurobi variables and the value they were assigned by the algorithm. Only those variables that have a value greater than 0 are stored in the solution.

The structure of the file is as follows:

```
<File> =
        <Version marker>
        <Solution statistics>
        <Gurobi parameters>
        <Heuristic parameters>
        <Solution name>
        FlowMatrix <Dimensions>
        <Flow matrix>
        DistanceMatrix <Dimensions>
        <Distance matrix>
        CapacityMatrix <Dimensions>
        <Capacity matrix>
```

```
        <Cost model>
        <Capacity model>
        <Gurobi solution>

<Version marker> = v<double>

<Solution statistics> =
        lowerBound '=' <double>
        objective '=' <double>
        computationTime '=' <double>

<Gurobi parameters> =
        nodeFileStart '=' <double>
        MILPFocus '=' <integer>
        presolve '=' <integer>
        timeLimit '=' <integer>
        MILPGap '=' <double>
        cuts '=' <integer>
        improveStartTime '=' <integer>
        improveStartGap '=' <double>
        MILPStartLocation '=' <string>
        workingDirectory '=' <string>

<Heuristic parameters> =
        tMin '=' <double>
        tFull '=' <double>

<Solution name> = <string>

<Dimensions> = <integer> <integer>

<Flow matrix> = <n x m matrix of integers>

<Distance matrix> =
        <n x m matrix of integers>
        <n x integer>
        <m x integer>

<Capacity matrix> =
        <n x m matrix of integers>
        <n x integer>
        <m x integer>

<Cost model> =
```

```
        hourFixedCost '=' <double>
        hourVariableCost '=' <double>
        kmFixedCost '=' <double>
        kmVariableCost '=' <double>
        roadHourCost '=' <double>
        roadKmCost '=' <double>
        roadSpeed '=' <integer>
        oneDayDistance '=' <integer>
        twoDayDistance '=' <integer>
        oneDayHubDistance '=' <integer>
        twoDayHubDistance '=' <integer>
        manyDaySpeed '=' <integer>
        hubTime '=' <double>
        transferCost '=' <double>
        pphCost '=' <double>

<Capacity model> =
        hubCapacity '=' <integer>
        trainCapacity '=' <integer>
        maxNumberOfBatches '=' <integer>

<Gurobi solution> =
        (<variable name> <double> '\n')*
```

C.3.3 MIP-start file

The MIP-start file is the file used by Gurobi to read an initial solution from at the start of the optimization process. This file is a simple, plaintext file in the format '<variable name> <value>'. These variable names need to match with the variable names in the problem to be solved, but need not be complete. Variables that are not assigned in this MIP-start file are solved for by Gurobi using the same procedure as used when no MIP-start file is specified.

C.4 Discussion

The requirements, package diagram, and file structure specifications provide some insight in the design choices that have been made while creating the tool. Important extension points for the tool are the classes in the UI package to add new features to the interface, and the Algorithms package in which new algorithms can be placed for inclusion in the tool. Other packages have been set up with further extension of features in mind.

The Development Tools package provides a generator for generating new test files and an algorithm runner which can use this generator to test the performance of the algorithm. The runner provides a batch processing feature

which is useful for testing the performance of the included algorithms, and may be extended to test new algorithms.