Optimising routing in an agent-centric synchronomodal network with shared information

Myrte De Juncker

Supervisors:
Sem Borst, TU/e
Frank Phillipson, TNO
Alex Sangers, TNO

Eindhoven University of Technology
Department of Mathematics and Computer Science

The Netherlands
August 2017
Abstract

This thesis is part of the NWO research project: “Complexity Methods for Predictive Synchronomodality” [140]. This program is aimed at innovative and multidisciplinary research to create new ideas for complex logistic systems. The complexity of the systems arises from the interaction between subsystems and the variability in the systems themselves. The research for this project has been done at TNO, in the department Cyber Security and Robustness. The mission of TNO is to connect people and knowledge to create innovations that boost the competitive strength of industry and the well-being of society in a sustainable way.

Our research focuses on synchronomodal planning problems. A synchronomodal system is a multimodal network in which logistic service providers have the opportunity to switch mode of transport of containers based on real-time data. This means that customers need to book amodally and the information within this multimodal network is shared to all agents. The planning problems that arise due to this added flexibility are synchronomodal planning problems.

First, we perform an extensive literature study that describes research concerning synchronomodal planning problems. We divide this research into strategic, tactical and operational planning. Research on operational planning is scarce, therefore our research focuses on this aspect. For synchronomodal planning problems a framework categorising this research does not yet exist and therefore we will develop such a framework.

Furthermore, we develop two different models of synchronomodal systems. We focus on a synchronomodal transportation system in which information is shared between all agents in the system. Here all agents are “selfish”, i.e. choosing routes based on an individual optimisation objective. Information can be public, i.e. known to everybody in the network or private, i.e. information that belongs to a certain agent which needs to be willing to share this information.

One of our models is an analytic model that relies on certain assumptions. This model is static in nature and therefore, not all aspects from real life cases are encompassed in this model. However, even with simplifications the analytic model is too computationally heavy. The other model is a simulation-based model, which can handle more realistic instances.

We also develop four different methods to determine the optimal paths in such an agent-centric synchronomodal network. We develop two simulation-based heuristics and two other simulation-based solution algorithms. The heuristics only act on public information, i.e. information about the congestion on the roads. The simulation-based models also act on private information, i.e. information about upcoming orders.

The two heuristics and one of the simulation-based methods are then tested on a synchronomodal network and on two smaller examples. The small examples show that there are instances in which the simulation-based solution algorithm outperforms the two heuristics. This means that the addition of private information can decrease costs. However, in the large synchronomodal network the differences between the different methods vanish and each model determines routes that result in similar costs. This means that the type of network and the network parameters determine whether the additional information is useful.
Contents

Abstract 1

Contents 5

I Introduction 6

1 Motivation and contribution 7

1.1 Motivation ......................................................... 7

1.2 Contribution .................................................. 13

1.3 Overview of the report ........................................... 14

II Synchromodal transportation in the literature 16

2 Literature review 17

2.1 Introductory papers .................................................. 18

2.1.1 Overview of synchromodal transportation .................. 19

2.1.2 Incentives towards synchromodality ....................... 19

2.1.3 Physical Internet ............................................. 20

2.1.4 Intermodal and unimodal transportation ..................... 20

2.2 Strategic planning problems ..................................... 20

2.2.1 Network structures .......................................... 21
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.2</td>
</tr>
<tr>
<td>Hub location problems</td>
</tr>
<tr>
<td>2.3</td>
</tr>
<tr>
<td>Tactical planning problems</td>
</tr>
<tr>
<td>2.3.1</td>
</tr>
<tr>
<td>Tactical planning problems in intermodal networks</td>
</tr>
<tr>
<td>2.3.2</td>
</tr>
<tr>
<td>Five practical planning problems</td>
</tr>
<tr>
<td>2.3.3</td>
</tr>
<tr>
<td>Synchromodality in tactical planning problems</td>
</tr>
<tr>
<td>2.4</td>
</tr>
<tr>
<td>Operational planning problems</td>
</tr>
<tr>
<td>2.4.1</td>
</tr>
<tr>
<td>Operational planning problems in intermodal networks</td>
</tr>
<tr>
<td>2.4.2</td>
</tr>
<tr>
<td>Synchromodality in operational planning problems</td>
</tr>
<tr>
<td>2.5</td>
</tr>
<tr>
<td>Our research and its relation to existing literature</td>
</tr>
</tbody>
</table>

| 3       |
| Framework of synchromodal transport planning problems | 30 |
| 3.1     |
| Motivation | 30 |
| 3.2     |
| Framework identifiers and elements | 32 |
| 3.2.1   |
| Identifiers | 33 |
| 3.2.2   |
| Elements | 34 |
| 3.3     |
| Notation | 35 |
| 3.4     |
| Examples | 38 |
| 3.5     |
| Discussion | 41 |

| III     |
| Optimal routing of containers in synchromodal transportation networks | 43 |

| 4       |
| Problem description | 44 |
| 4.1     |
| Description of the research | 44 |
| 4.1.1   |
| Research questions | 45 |
| 4.2     |
| Approach | 45 |
5 Review of Dynamic Traffic Assignment and information availability in traffic systems 47
   5.1 Dynamic Traffic Assignment ........................................ 47
   5.2 Information availability in traffic systems ...................... 50
   5.3 Research used for expansion in synchronodal systems .......... 51

6 Multi-Class Open Product-Form Networks 52
   6.1 Theory of Multi-Class Open Product-Form Networks ............ 53
   6.2 Modelling of a traffic network ................................... 56
   6.3 Modelling of an intermodal network ............................ 63

7 Simulation-based models 67
   7.1 Assumptions ....................................................... 67
   7.2 Description of simulation ........................................ 68
   7.3 State-dependent travel times .................................... 70
   7.4 Simulation-based heuristics ..................................... 72
      7.4.1 Heuristic 1: minimum-cost routing without rerouting .... 73
      7.4.2 Heuristic 2: minimum-cost routing with rerouting ....... 73
      7.4.3 Dynamic shortest-path algorithm .......................... 74
   7.5 Simulation-based solution algorithm ............................ 78
      7.5.1 Solution for user equilibrium routing ...................... 79
      7.5.2 Rolling-horizon framework .................................. 86
   7.6 Discussion ....................................................... 87

8 Results 89
   8.1 Synchronodal network and parameters ............................ 89
      8.1.1 Trans-European Transport Network .......................... 89
      8.1.2 Parameters of the Trans-European Transport Network .... 91
   8.2 Results of the models and routing methods ...................... 93
      8.2.1 Scenario analysis ............................................ 93
      8.2.2 Results on the TENT network ................................ 97
      8.2.3 Notes on the analytic model ................................ 99
      8.2.4 On the solution algorithm within the rolling-horizon framework 100
9 Conclusions and discussion

9.1 Conclusions ................................................. 101

9.2 Discussion and further work ........................................ 104
  9.2.1 Results on real life networks .................................. 104
  9.2.2 Results on the analytic model .................................. 105
  9.2.3 Results on the rolling-horizon framework ......................... 105
  9.2.4 System optimal routing ......................................... 105

Bibliography

URL references ................................................ 113

Internal documentation and lecture notes ................................ 114
Part I

Introduction
Chapter 1

Motivation and contribution

This chapter starts with some introduction and motivation for this project, Section 1.1. Here we will explain important terms and why this research is relevant. In Section 1.2 we elaborate on the details of our research. We end this chapter with Section 1.3, which gives an overview of the rest of the report.

1.1 Motivation

TNO is an independent research organisation. Their mission is to connect people and knowledge to create innovations that boost the competitive strength of industry and the well-being of society in a sustainable way. TNO focuses on the themes: Industry, Healthy Living, Defence, Safety & Security, Urbanisation and Energy. The department Cyber Security & Robustness helps their customers to maintain security and business continuity. Their research helps with the integrity, availability and confidentiality of databases and networks. They do research projects for the government and for companies and can also help with the development of security policies. They do (among other projects) risk analysis and security assessments for government agencies. They also work together with universities for projects throughout the EU. The department also researches robustness and performance of networks, largely in telecom networks. Since the research in telecom networks has large overlap with the research in other networks, the department also collaborates with other departments and companies in the logistics sector.

For the next five years, TNO cooperates with universities in a proposal for the call ‘Complexity in Transport and Logistics’ by the NWO [140]. This program is aimed at innovatory and multidisciplinary research to create new ideas for complex logistic systems. The complexity of the system is in the interaction between subsystems and the variability in the system itself. Understanding this complexity will make sure that we can recognise certain behaviour and maybe even influence these systems. One of the items in this proposal is the use of Stochastic Assignment Problems for planning in a synchromodal system. We will elaborate on these terms further down this report. This thesis will be a part of this bigger project.

In this thesis we study stochastic centralised networks with as use case: container transport. The most widely used method nowadays to transport goods from one place to another is intermodal transportation. Most of the containers that are shipped have a standardised size. This allows the containers to be shipped with different modes of transport. Therefore, goods may start on a truck, to be placed on a barge and/or a train from where the last part of the journey can be by truck again.
The current planning of these containers has some disadvantages. Often, the route is planned beforehand. Thus, if one of the modes (barge or train) is delayed, the goods are be placed on a truck to get to their final destination on time. However, a truck is not efficient in terms of cost and CO\textsubscript{2} and there might be possibilities to use other modes. Another issue is the collaboration between all different parties. If multiple parties would work together and group their containers on a ship, instead of all using their own ships, transportation may get much more sustainable.

In order to make container transportation more efficient, reliable and sustainable, we are investigating \textit{synchromodal transport planning}. This allows a logistics service provider to schedule transport according to what currently is the best available transport mode. For example: a container travels on one day by train on a part of its route, but another day a container on the same route and the same part travels by barge. In synchromodal transportation planning two things are important [151]:

1. Customers will only tell the logistics service provider when and where their cargo needs to arrive, therefore, entrusting the logistics service provider with the planning.
2. Planners will use data that is real-time, and routes will become subject to change in real-time when beneficial. This will be in terms of time, costs and CO\textsubscript{2} emissions.

We will focus on the second item. In most approaches to the planning of synchromodal transport, planning is based on the information on a certain moment. This means that with every new event a new, possibly totally different, planning has to be made. New planning solutions need robust or stochastic optimisation, as in [101] and [76].

Since synchromodal transport is a relatively new concept within the logistics sector a lot needs to be investigated. There are a lot of problems that need to be addressed before synchromodal transport can be the new standard of transporting goods.

The paper by Pfoer, Treiblmaier and Schauer [102] defines seven critical success factors of synchromodality. A successful implementation of a synchromodal transport chain only works when all factors are addressed:

- \textit{Network, collaboration and trust}. All agents in the synchromodal system have to cooperate. Many companies are reluctant to cooperate with competitors, but they have to see the potential of the synchromodal system.
- \textit{Sophisticated planning}. The planning of a synchromodal network is complex. Firstly, one has to plan the services, i.e. trucks, trains and barges. Then for the containers there are different modes that can be chosen and the planning has to be adapted in real-time.
- \textit{ICT/ITS technologies}. The sharing of data is critical in a synchromodal network. Therefore, there have to be good ICT and ITS systems that provide data in real-time.
- \textit{Physical infrastructure}. One also has to look at the terminals (places where a container can switch modes) in a synchromodal network. The placements and design of these terminals affects the efficiency of the network.
- \textit{Legal and political framework}. The liability of the transport is not as clear as in an intermodal network. Who is responsible for delay, loss or damage? For these kind of questions a legal and political framework is needed so all parties have legal security.
• **Awareness and mental shift in customers.** The customers also have to be willing to ship their containers through a synchromodal network. They have to trust the logistics service provider with most decisions. The customers need to know the advantages of synchromodal transportation before they have this trust.

• **Pricing, cost and service.** The last factor is pricing, cost and service. If transport in a synchromodal network is not cheaper or at least the same price as in an intermodal network, customers will always choose an intermodal network. The pricing is complex, because the modes and routes are not determined beforehand.

We are going to look at a part of the sophisticated planning mentioned above. The real-time data of a synchromodal system may introduce new problems. With real-time data, goods can be efficiently rerouted, but it may be that rerouting a certain container means that a lot of other containers will be rerouted because of this action. This might work contra-productive for the entire system.

When reviewing the planning of a synchromodal system, a lot of choices have to made. For instance, the schedules of the resources and the planning of the containers to these resources. In synchromodal transport an agent (e.g. container, truck, barge, train, port, etc.) needs to take a decision on the assignment of containers to one of the modes of transportation. It depends on the system how this decision is taken. As described in [152], we have to look at two aspects: information and the degree of control and optimisation. Both can take either a local view, where only own information is known and optimisation is for an individual objective. Or a global view, where information is available for the entire network and the optimisation is aimed at a shared goal. We can distinguish four different systems in a framework, see Figure 1.1.

![Figure 1.1: Framework, from [152].](image)

If the information is available globally but every agent only optimises their own objective, we call the approach *selfish*. If the information is available globally and the decision is aimed to optimise the entire
network, it is called social. If the information is only available locally and optimisation is also local, it is a limited approach. Lastly, if the decision is aimed at global optimisation with local information, we call it a cooperative approach.

Information is a broad term, some of the information is public, which means every agent can get this information. Other information is private and has to be shared between different stakeholders in the network. This sharing can be difficult to achieve, since the stakeholders need to be willing to share their private information to competitors and clients. Therefore, we can add another level to the framework, namely the type of information that is available, see Figure 1.2.

Figure 1.2: Enhanced framework, from [152].

In this project, we will look at systems that fall in the upper left corner. This means that we look at agent-centric or selfish systems with varying degrees of information. We want to investigate how the inclusion of private information can benefit individual optimisation objectives. These systems are represented by networks of nodes and edges as explained below.

Networks in transportation planning

Networks are an important tool in transportation planning. A network consists of nodes and edges. Here the nodes define places from where and to where the containers travel. The edges are defined as the roads from one node to another. One can think of the map of for instance the London subway, see Figure 1.3. Here all nodes are the places that the subway stops and the lines between the destination nodes are the edges of the transportation network.
Figure 1.3: A transport network, from [141].

If one uses only one mode of transport, the network stays relatively simple. Between all nodes that have a connection a single edge is drawn. However, in intermodal and thus synchromodal transport, there can be multiple edges between nodes. These edges represent the different modes of transport that are available. For instance, from Rotterdam to Hengelo, containers can be transported by truck and by barge. This means that in the transportation network, there will be two edges connecting Rotterdam to Hengelo. One stands for the road option and one for the waterway option. To give an impression of such a network, see Figure 1.4. Figure 1.4 is a map (zoomed in) of the Trans-European Transport Network (TEN-T) and specifically the North Sea - Mediterranean corridor. This network is maintained and improved by the European Commission.
In Figure 1.4 we can see multiple links between certain cities. This means that the containers travelling between these cities can be transported by different modes.
1.2 Contribution

This section elaborates in more detail the research that has been done in this project. We start by explaining the goals we want to achieve, followed by the modelling of the problem. Finally, we discuss what kind of results we hope to achieve.

Goals of the project  As discussed before, in Section 1.1, we mainly focus on the information availability in an intermodal network. Global information availability is a step towards a synchronodal system. We focus on agent-centric networks, where each agent is selfish and wants to optimise their own objective function. In real life situations a lot of stakeholders are present. Each of these stakeholders has their own agenda and wants to optimise their own objective function. This resembles an agent-centric network.

We distinguish two types of available information: public information and private information. Public information is information that is available for everyone. This means that for public information we do not need different stakeholders to cooperate. The public information we encounter in our problem is information about the occupancy of different links. This means that we know, for any point in time, how many agents are on every link of the network. One can already see examples of this public information being used for road networks. For example: route guiding systems already have an option to recalculate your shortest path based on information about congestion in the network. Private information on the other hand, does require different stakeholders to cooperate. The private information we encounter is information about upcoming orders. Different logistic service providers normally know what orders are going to arrive in the near future. If these logistic service providers would be willing to share this information with all stakeholders, all stakeholders can react upon this information. This means that agents also have information about the future occupancy of certain links.

This project investigates the effects on the costs and thus on the optimisation objectives of all agents with different information availability. This means that we assume cases where only public information is available and cases where agents have access to both public and private information. We develop five different models for the analysis. We will investigate the similarities and differences between the different models.

Modelling  We construct two different models for an agent-centric intermodal network with shared information. We develop one analytic model and one simulation-based model. The simulation-based model is capable of analysing more realistic instances.

- Analytic model: the first model is an analytic model based on stochastic networks. We modelled the agent-centric network as a Multi-Class Open Product-Form Network. Here each link in the network is represented with different queuing models, that aim to capture the relevant processes of containers moving over a road, rail, or waterway.

- Simulation-based model: the other model is an event driven simulation, that simulates the agents going through the synchronodal network.

For the simulation-based model we developed four different methods to generate routes for the containers. All methods aim to minimise certain costs. The four methods are described below:
- **Heuristic 1**: The first heuristic is a naive implementation with public information. We assume that each agent checks the public information at departure and will react accordingly. This means they will take the shortest path for the current state of the network. After the choice is made, they will not deviate.

- **Heuristic 2**: The second heuristic is very similar to the first, except the fact that agents do switch routes before reaching their destination. This means that at each decision point, i.e. an intermediate node in the network, they again check the state of the network and reroute if necessary.

- **Simulation-based solution method**: the simulation-based solution method assumes full information. This means that this algorithm assumes that each agent knows exactly what is going to happen for a certain planning horizon. This includes public as well as private information. With this information, the algorithm seeks to find the optimal routes for all agents that arrive somewhere in the planning horizon.

- **Simulation-based solution method in a rolling-horizon framework**: Private information will not always be accurate. Therefore, we also developed a model that assumes that for a short time period reliable estimates are known and for a longer time period estimates are known that are less reliable. This algorithm relies heavily on the simulation-based solution method.

**Results** In Chapter 8 we discuss the results of our analysis. Unfortunately, real data was not available for this project. Therefore, we analyse the Trans-European Transport Network with self-chosen parameters, see Figure 1.4. We aim to model real life instances with our chosen parameters. For this network we want to know the effects of increasing the amount of information on the objective functions for the agents. One would expect that having access to more and more information increases the possibility to take an optimal route.

We are also interested in the similarities and differences between our different models. The analytic model is restrictive, since it is a static network. This means that all the dynamic aspects found in transportation networks are not captured inside the model. The simulation-based models, on the other hand, do take into account the dynamic aspect. It will be interesting to study the difference between the heuristics and the simulation-based solution method. The heuristics are easily implemented for real life instances and do not require private information. If these heuristics perform nearly as good as the simulation-based solution method, one has to wonder if it is necessary to include all information in the routing of containers. On the other hand, if the simulation-based models outperform the heuristics, it is a good incentive for agents in intermodal networks to start to cooperate. This might be beneficial for all of them.

**1.3 Overview of the report**

Part II of this thesis, describes previous research on intermodal and synchronmodal networks. In Chapter 2 we give an overview of the literature. As said before, this thesis is part of a larger project and therefore the literature review is quite extensive. We did not only focus on the aspects of information in an agent-centric network, but we looked at different planning problems on different time horizons, long-, medium- and short-term. During this literature review we found that a framework for synchronmodal planning problems was missing. Therefore, in Chapter 3, we develop a framework that tries to capture the most relevant modelling choices in synchronmodal systems.
Part III focuses on the information availability in a agent-centric intermodal network. Here, we first mention current literature on Dynamic Traffic Assignment in Chapter 5, since road networks are comparable to transportation networks. From the approaches in the literature on traffic assignment we gather approaches to the routing of containers on an intermodal network.

In part III we mention current literature in Dynamic Traffic Assignment and related subjects in Chapter 5. Thereafter, we model a road network and synchromodal network analytically as well as based on a simulation. In Chapter 6, the analytic analysis can be found and in Chapter 7 we describe the simulation-tool that analyses the same systems and the four simulation-based models. Lastly, in Chapter 8 we discuss the results we found on the Trans-European Transport Network and the differences and similarities between our models. In the final chapter, Chapter 9, we derive the conclusions of our work and mention further research that can be done to extend the work done in this thesis.
Part II

Synchromodal transportation in the literature
Chapter 2

Literature review

We start this chapter with a note about the different terminology that is used throughout the literature. In Chapter 1 we defined *synchromodal transportation* as a network in which customers entrust the logistics service provider by only sharing when and where their cargo needs to arrive and where real-time data is used to reroute containers if necessary. We also mentioned *intermodal transportation* as a type of transport in which containers are used to transport goods via different modes. Multiple terms are used in the literature. Since not all definitions are the same, it is hard to make a clear distinction between the different terms. We will try to give the distinction between these terms below.

- **Transport links:** the terms below are used to describe how the goods within a network are transported.
  - Unimodal transportation: if cargo is transported by only one type of transport (e.g. truck, train or barge) we call it unimodal transportation.
  - Multimodal transportation: this term is used if cargo is transported by using multiple types of transport, i.e. truck and train are combined to transport the cargo.
  - Intermodal transportation: intermodal transportation differs from multimodal transportation by the fact that in intermodal transportation all goods are *containerised*. In multimodal transportation this does not have to be the case.

- **Network characteristics:** the terms below are about characteristics of the entire network.
  - Co-modal transportation: co-modal transportation refers to the use of two or more modes of transport on their own *and* in combination to get the biggest benefit from each of them.
  - Synchronodal transportation: this is the type of network that we focus on. Here real-time data is used to reroute containers and information is shared between all agents in the network.

In this chapter, we give an overview of the relevant literature on synchronodal transportation. However, this research area is fairly new and has a lot of similarities with other intermodal transportation problems. Therefore, we also add literature about intermodal transportation. From this literature review we want to create a sense of the problems that arise when looking at synchronodal transportation. We focus this review on the operations research in synchronodal networks. For a more extensive literature review, we refer to SteadieSeifi et al. [121]. This paper includes synchronodal transportation and focuses on papers...
from 2005 until 2013. For an older literature review focusing only on intermodal transportation, we refer to Macharis and Bontekoning [81].

Even when focusing on operations research, one can still make a distinction in the focus of the problems. This is captured in the 4S framework of Rudy Negenborn [139], shown in Figure 2.1. It shows that we need smart equipment that works in smart hubs for smart ports in smart networks. We focus on problems involving the entire network, i.e. a network of terminals, transportation modes and containers. One could also look at the planning problems of, for instance, the multimodal terminals (places where the containers can change mode of transport, i.e. from truck to train or from barge to train). Specific terminal problems are: buying handling equipment, planning human resources and allocation berths (the places where the ships can unload and load the containers). We refer to the review paper by Steenken et al. [122] and its update by Stahlbock and Voß [120] for the planning problems within a terminal.

Within operations research questions, papers mainly focus on long-, medium- or short-term planning problems. We refer to these problems as strategic, tactical and operational planning problems, respectively.

In this chapter, we start with some introductory papers in Section 2.1, that deal with synchromodal transportation as a whole. Afterwards we divide the papers under the strategic, tactical and operational planning in Sections 2.2, 2.3 and 2.4. In the final section of this chapter, Section 2.5, we will mention what gaps we see in the literature on synchromodal transportation and where the research in this report fits in.

2.1 Introductory papers

In this section we mention papers with an overview about synchromodal networks in Section 2.1.1. In Section 2.1.2 we mention some incentives that can be given to ensure coordination between different agents in a synchromodal network. In Section 2.1.3 we mention parallel research about Physical Internet. This research has a lot of overlap with research in synchromodal networks. In the last section, Section 2.1.4, we mention some papers that give overviews in intermodal and unimodal transportation and their most common problems.
2.1.1 Overview of synchromodal transportation

The paper by Tavasszy, Behdani and Konings [124] is an introduction in synchromodality. They discuss the current position and evolution of intermodal transportation, the main elements of the synchromodal transport chain and the innovations that are necessary to arrive at synchromodal transportation systems. They describe the growing trend in freight transportation and the need for an efficient organisation of hinterland transport services. The main element of a synchromodal transportation that is mentioned in this paper is the integration of transport service on different modalities with real-time availability of the services. Changes have to be made to the network in order to create a synchromodal system. Among others, there is need for an integrated network and service design, an integrated operation and control, contracts that allow synchronised transport, a stronger collaboration and a mind shift in planning and control.

Van Riessen, Negenborn and Dekker [113] also give an overview of current topics and research opportunities in synchromodal container transportation. They focus on the case of the hinterland network of European Gateway Services and thus provide no complete overview. The paper has three main topics: optimisation of integral network planning, methods for real-time decision making and the creation of flexibility in the network planning problem.

The paper by Pfoser, Treiblmaier and Schauer [102] determines the critical success factors of synchromodality as mentioned in Chapter 1. They come up with a list of seven factors: Cooperation and Trust, Sophisticated Planning, ICT/ITS Technologies, Physical Infrastructure, Legal and Political Framework, Awareness under Customers, and Pricing, Cost and Service.

2.1.2 Incentives towards synchromodality

In transportation networks there are a lot of stakeholders that all may have a different notion about optimality and generally make decisions on their own. In synchromodal networks all these stakeholders need to work together. This coordination does not come naturally. The paper by Van Der Horst and De Langen [65] describes the problems that arise during this coordination. They also present four main categories of arrangements that can be made to improve coordination:

1. Introduction of incentives: an operator can choose certain incentives that will maybe give other agents reasons to cooperate. One can for instance think about a bonus or penalty for certain agents or the auction of capacity.

2. Creation of an interfirm alliance: an alliance between companies is an arrangement with more commitment than incentives. Here we think of subcontracting, standardised procedures or a joint capacity pool.

3. Changing scope: changing the scope of the organisation is even one step further than the creation of interfirm alliances. One could introduce a new agent or chain manager or for instance look at a whole new market.

4. Creating collective action: the final category is collective action. This is about investments that have collective benefits. These can be structured through a public organisation or an industry association. Among others, we can think of branch association or an ICT system for a sector of industry.
2.1.3 Physical Internet

A lot of research on synchromodality is done in The Netherlands. This is because The Netherlands is a major player in the transportation of goods. Goods enter The Netherlands via the port of Rotterdam from where they are shipped throughout Europe. However, research into synchromodal transport is not limited to The Netherlands. In France, the same concepts are studied under a different term: Physical Internet. The paper of Montreuil [90] introduces the term Physical Internet and its key characteristics.

The idea of the Physical Internet is largely the same as the idea of synchromodal transportation, namely to make transportation more reliable and sustainable. However, there are also some key differences. Literature about the operations research problems within the Physical Internet may also be of interest, but papers introducing the Physical Internet also mention the design of world-standard smart green modular containers and the problem of fitting products in the containers with minimal space waste. These problems are out of scope for this report. We refer to the paper of Montreuil [90] and the paper of Ballot, Goblet and Montreuil [15] for more literature regarding the Physical Internet.

2.1.4 Intermodal and unimodal transportation

As said before, synchromodal transportation is a fairly new topic within the scientific literature. Therefore, we mention some papers about intermodal and unimodal transportation that may be of interest, because of the similarities in the type of problems.

Bektas and Crainic [19] give an overview of the players in an intermodal network and the challenges they face. They also give a short description of some of the most common problems within an intermodal network and the contribution of operations research to this field. They look at the shipper perspectives on intermodal transportation, who have to decide on a certain transportation mode. They also mention the carrier perspective, who have to provide an efficient and cost effective service to the customer. This includes system design, service network design (making the transportation plan) and operational planning (day-to-day planning). Terminals are also discussed as one of the players in a transportation system, which have to handle the loading/unloading of containers between different modes.

The article by Crainic and Kim [43] gives a nice overview of all problems concerning intermodal transportation. They follow the same classical approach of examining issues, models and methods according to the strategic, tactical and operational level of planning and operations. They also discuss freight transportation and the main actors and issues.

Christiansen et al. [38] mention the operations research related problems (strategic, tactical and operational) of maritime transportation with an emphasis on ship routing and scheduling models. The term maritime transportation refers to transportation over seas, but they also include inland waterways in their paper.

2.2 Strategic planning problems

Strategic planning problems are long-term planning problems. In the context of intermodal and synchromodal transportation, they include decisions on the infrastructure and placements of terminals. In Section 2.2.1 we mention the different structures that are possible within intermodal transportation. In Section 2.2.2 we refer to literature about locating terminals within such a network.
2.2.1 Network structures

Firstly we elaborate on the different network structures that are possible within intermodal transportation. The terminology is based on the paper by Woxenius [135]. In this paper six different options for transportation are defined. Note that in real networks, combinations between the different structures can be used. In the paper, the author assumes that all nodes are capable of serving as origins and destinations as well as terminals, i.e. points where containers can be moved from one mode of transport to the next. The different structures are illustrated in Figure 2.2 and explained below.

- Direct link: the containers are transported from the origin to the destination directly. Therefore, no other nodes are involved.
- Corridor: in this design a path is formed that has a high-density flow. Nodes that are not along this path (called satellite nodes) have short links to the nodes that are on the path (called corridor nodes).
- Hub-and-spoke: here one node is assigned as a hub and all goods go through this hub to be transported to their destination.
- Connected hubs: in this design there are multiple nodes that are assigned as hubs. Local transport is collected at a hub and transported through other hubs to its destination.
- Static routes: the transport operator designates the links that have to be used on a regular basis. This means that the route is predetermined by an operator.
- Dynamic routes: this is the design with the most flexibility. The operator can choose between different routes from the origin to the destination and they may even change during transportation.

In the literature, hub-and-spoke networks and connected hub networks are studied the most, see the paper by SteadieSeifi et al. [121].
2.2.2 Hub location problems

The main problem in the strategic planning of a network with hubs is the placement of these hubs. This is called the **hub location problem**. The problem is concerned with locating hub facilities and allocating all nodes to these hubs. For an extensive classification and review of hub location models, we refer to the paper by Alumur and Bahar [4]. This paper classifies and reviews hub location models.

In a hub location problem, a network of \( n \) nodes is given. From this network we know which nodes are the origin, destination and possible hub location nodes. The most common assumptions for the hub location problem are: there is a link between every two hubs, there is a discount factor (\( \alpha \)) for using the hubs, and direct service, between two non-hub modes is allowed. We also know the flow between origin and destination nodes, the optimisation criterion and the discount factor (\( \alpha \)) before locating the hubs.

We classify the hub location problems as in [4] in four different problems. Each of these problems can be divided in two different types of networks, namely *single allocation* and *multiple allocation*. In single allocation, each non-hub node is allocated to exactly one hub. In contrast, in multiple allocation, non-hub nodes can send and receive flows from and to multiple hubs. The paper by Yaman [136] looks at \( r \)-allocation problems, where each node is allocated to at most \( r \) hubs.

1. **The \( p \)-hub median problem**
   In the \( p \)-hub median problem we want to minimise the total transportation cost. We are given \( n \) demand nodes, flows between origin-destination pairs and the number of hubs to locate (\( p \)).
   - Single allocation
     The first hub location problem described was a single allocation \( p \)-hub median problem. This problem is formulated in the paper by O’Kelly [94]. Skorin-Kapov et al. [117] present a mixed integer formulation which can be solved using CPLEX. Ernst and Krishnamoorthy [51] propose a different linear integer programming formulation in their paper, which reduces the previous formulation.
     The Ph.D. thesis of Kara [69] shows that this problem is NP-hard. The solution procedure by Ernst and Krishnamoorthy [52] can find optimal solutions for networks with up to 100 nodes. Since the problem is NP-hard a lot of heuristics are studied. The best heuristic found so far is by Pirkul and Schilling [104]. Other heuristics for this problem are studied in the papers by O’Kelly [94], Klincewicz [74] and [75], Skorin-Kapov and Skorin-Kapov [116], Campbell [28], Ernst and Krishnamoorthy [51], Smith et al. [118] and Abdinnour-Helm [1].
   - Multiple allocation
     Campbell [26] was the first to formulate the multiple allocation \( p \)-hub median problem as a linear integer program. The most efficient formulation is proposed in the paper by Ernst and Krishnamoorthy [53].
     Two papers by Ernst and Krishnamoorthy, [53] and [52], present a solution method to obtain exact solutions, but the algorithm from [52] outperforms the algorithm from [53]. Also for this problem, multiple heuristics are studied. We refer to the papers by Campbell [28] and Ernst and Krishnamoorthy [53].

2. **Hub location problem with fixed costs**
   In this problem the fixed costs of opening facilities are incorporated. This means that we still want to minimise total transportation cost with a given network of \( n \) nodes and given flows between the origin-destination pairs. However, the number of hubs to locate is a decision variable in this problem.
   Again we can look at single/multiple allocation problems. However, in this problem we also distinguish uncapacitated/capacitated hub location problems. For the capacitated problem, hubs have limited capacities.
• **Single allocation**
  O’Kelly [95] was the first to introduce the single allocation hub location problem with fixed costs. The first presentation of the capacitated version is in the paper by Aykin [11].
  - Uncapacitated: The paper by Abdinnour-Helm and Venkataramanan [2] formulates a quadratic integer program and solves this with a branch-and-bound procedure. Heuristics for this problem have also been extensively studied, among others in: Cunha and Silva [44] and Chen [35].
  - Capacitated: Ernst and Krishnamoorthy [54] present new formulations for this problem, a solution method to obtain optimal solutions and two heuristics.

• **Multiple allocation**
  - Uncapacitated: Campbell [27] presented the first linear programming formulation for the multiple allocation uncapacitated hub location problem. The solutions to this problem have been studied in (among others): Mayer and Wagner [83] and Cánovas et al. [29].
  - Capacitated: The capacitated multiple allocation hub location problem with fixed costs is studied by Ebery et al. [49].

3. **The $p$-hub centre problem**
The paper by Campbell [27] defines three different types of $p$-hub centre problems:

(a) The maximum cost for any origin-destination pair is minimised.
(b) The maximum cost for movement on any single link is minimised.
(c) The maximum cost for movement between a hub and an origin/destination is minimised.

• **Single allocation**
  Linear programming formulations for this problem can be found in the paper by Kara and Tansel [70] and Ernst et al. [55]. Heuristics are presented in the paper by Pamuk and Sepil [96] and the paper by Ernst et al. [55].

• **Multiple allocation**
  This problem is studied in the paper by Ernst et al. [55].

4. **Hub covering problems**
Demand nodes are considered covered in a hub covering problem if they are within a specified distance of a hub. Campbell [27] defines three coverage criteria for hubs, mentioned below. They also describe mixed integer programming formulations for these problems.
The origin destination pair $(i,j)$ is covered by hubs $k$ and $m$ if:

(a) the cost from $i$ to $j$ via $k$ and $m$ does not exceed a specified value.
(b) the cost for each link in the path from $i$ to $j$ via $k$ and $m$ does not exceed a specified value.
(c) each of the origin-hub and hub-destination links meets separate specified values.

• **Single allocation**
  Kara and Tansel [71] study this problem and prove it is NP-hard.

• **Multiple allocation**
  We have found no papers studying the multiple allocation hub covering problem.

All hub location problems mentioned above are discrete. This means we already have some possibilities for the location of the hubs and we want to know which are the best. One could also look at the problem of placing hubs on a plane. For this problem we refer to the paper by Aykin and Brown [12]. We will not go into detail for this problem, since we look at networks where the locations are predetermined.
2.3 Tactical planning problems

In this section we mention the tactical planning problems within intermodal transportation in Section 2.3.1. In Section 2.3.2 we mention five practical cases that are studied in the literature. In Section 2.3.3 we elaborate on some papers that explicitly deal with synchromodality.

2.3.1 Tactical planning problems in intermodal networks

The tactical planning problems within intermodal transportation are about the planning of the services. Thus choosing, for instance, which train will go on which track and how often. One also has to think about the pricing of the services. We will not elaborate on this aspect of the system. The interested reader is referred to the dissertation of Tsai [131]. The planning of the orders, i.e. which goods will be transported by which mode on which time, also fall under the tactical planning problems. However, dealing with the day-to-day operations of the order planning is operational planning. Some papers and problems, therefore, fall under tactical and operational planning. We try to make the division as clear as possible. Problems that deal with real-time requirements will be under operational planning and all others under tactical planning.

The tactical planning problem is quite extensive. One needs to select and schedule the services to operate, allocate the capacity and equipment, and look at the routing of the goods. Together this is also called Service Network Design. The review paper of Crainic [39] gives an extensive review of these problems, their formulations and their solution frameworks. The author also gives a classification of these problems.

The main issues at tactical level in an intermodal network are:

1. Service selection: the characteristics of the services that are offered in the network (how often, when, what capacity etc.).
2. Traffic distribution: the distribution of the goods through the network (which goods go with which service).
3. Terminal policies: rules that specify for each terminal how to perform the consolidation activities.
4. Assignment of crews and vehicles: how to assign crew and vehicles to the planning.
5. Empty balancing strategies: how to reposition empty vehicles.

In the literature these problems are mostly modelled as Fixed-Cost Capacitated Multicommodity Network Design Problems. These formulations are defined on graphs containing nodes (or vertices) that are connected by links. These links are typically directed and represented by arcs. The nodes are demand and destination points for the goods. The links have certain characteristics such as length, capacity and cost. Some of these costs can be fixed costs, meaning that if one chooses that link, the fixed costs are incurred, no matter how much goods travel over that link. One wants to choose the links and capacities in the network, such that the demand is satisfied at the lowest possible cost. What defines a cost function is not always the same. Some models look only at cost, while others also take into account service levels. The paper by Min [89] develops a chance-constrained goal programming model that has multiple aspects in the objective function.

When schedules are planned, a time dimension must be introduced into this problem, making it a (discrete) multi-period model. To represent this problems, space-time networks are used. Figure 2.3
shows an example of a space-time network. In a space-time network all terminals are modelled as nodes. Services go from these terminals to other terminals in a certain amount of time. These services are modelled by the arcs. One such arc can, for example, be a barge going from a terminal in Rotterdam (node 1) to Hengelo (node 3) in one time period.

Figure 2.3: Example of a space-time network and an example of a feasible service plan, from [7].

To solve these tactical planning problems, constraints can be relaxed to obtain a simpler problem. This yields lower bounds that together with heuristically found upper bounds can be combined in for instance a branch-and-bound algorithm, see, among others, the paper by Crainic et al. [40] and Holmberg and Yuan [64]. Holmberg and Hellstrand [63] propose an exact solution method for the uncapacitated problem based on a Lagrangian heuristic. A dual ascent procedure is treated by Balakrishnan et al. [14], which finds lower bounds within 1 – 4% of optimality. Heuristics and meta-heuristics (such as Tabu Search, Simulated Annealing and Genetic Algorithms) are also widely used. See for instance the paper by Crainic et al. [42], who look at a path-based formulation of the same problem and solve it with tabu search. Other papers looking into these meta-heuristics are among others: Bai et al. [13], Chouman and Crainic [37] and Pedersen et al. [97].

Andersen and Christiansen [5] present a service network design model that is applied to a case study involving a rail infrastructure. They illustrate the problem with a space-time network. The decisions in the problem are on how many vehicles are to be employed, which services to operate and how often, and towards which demand potential to direct the services. In this paper they also look at synchronisation within the system and with the neighbouring systems. In another paper, Andersen et al. [6] look at a service network design with asset management. Here they integrate the management of assets, i.e. vehicles, power units, etc. A faster solution algorithm based on a branch-and-price algorithm for this problem is proposed in the paper by Andersen et al. [8].

The papers by Newman and Yano [92] and [93] look at train scheduling in an intermodal network. In the first paper ([92]) they look at the difference of centralised versus decentralised decision-making. In the second paper ([93]) they focus on determining day-of-week schedules for both direct and indirect trains and allocating containers to these trains.

All models discussed above do not look at the uncertainty that is, unavoidable, in an intermodal system. However, there are a few papers that do take into account uncertainty. From these papers, most look at the case in which the demand is stochastic or uncertain. Lium, Crainic and Wallace [80] look at the differences in solutions that are based on uncertain demand or deterministic demand. The paper by Hoff et al. [62] look at the time-dependent service network design problem with stochastic demand. They represent their problem by a space-time network and find solutions by combining exact and heuristic
methods. Finally, the paper by Meng, Wang and Wang [85] looks at a short-term ship fleet planning problem with uncertain container shipment demand. They model the problem as a two-stage stochastic integer program and propose a solution algorithm based on dual decomposition and Lagrangian relaxation. A paper by Kooiman, Phillipson and Sangers [76] looks into a case study of planning inland container shipping. They propose a simulation algorithm which maximises the number of containers arriving on time, transported by barge. In this paper the containers have a stochastic release time. This means that it is uncertain if the container is available on time at a certain port.

2.3.2 Five practical planning problems

The paper by Wieberneit [133] focuses on five practical planning problems, which are extensively described with solution methods. We mention them below only briefly. As tactical planning problems are extensive, it is common to search for solution methods that are tailored for a specific instance.

- The paper by Barnhart et al. [17] analyses the Express Shipment Delivery Problem of UPS. They propose an iterative approach based on conventional network design methods. The papers by Armacost et al. [9] and [10] develop a composite variable formulation for the same kind of problem.

- The Lettermail Flight Network is a tactical planning problem of the Deutsche Post AG. The paper by Büdenbender et al. [25] describes a branch-and-bound/hybrid tabu-search algorithm for this problem.

- Another problem is the Less-than-truckload operations in North America. For this problem different modelling and solution approaches have been proposed. Powell and Sheffi [106] develop a local improvement heuristic, which is extended by Powell [105]. A decomposition strategy is proposed in the paper of Powell and Sheffi [107]. Farvolden and Powell [56] present a subgradient method.

- Jansen et al. [67] consider a multimodal transportation system from the Deutsche Post World Net in Germany, Less-than truckload operations on a Multimodal Network in Europe. They implement a decomposition algorithm with six sub-problems.

- The last problem is Less-than-truckload operations on a Road Network in Europe. Wlcek [134] (book in German) considers a road network in Germany and proposes a sequential decomposition approach.

2.3.3 Synchromodality in tactical planning problems

The paper by Puettmann and Stadtler [108] mentions the importance of coordination of plans of independent service providers in an intermodal transportation chain. They present a coordination scheme that will lead to reductions in overall transportation costs. They include stochastic demand in their calculation of the overall costs. Another paper by Caris, Macharis and Janssens [31] also looks at cooperation between inland terminals. In the paper they develop a service network design model for intermodal barge transport and apply it to the hinterland network of the port of Antwerp. They simulate cooperation schemes to attain economies of scale. The paper by Behdani et al. [18] develops a mathematical model for a synchromodal service schedule. Taking into account the frequency and capacity of different modalities, it determines the optimal schedule and timing of services for all transport modes. The assignment of containers to services is also determined by the model.
2.4 Operational planning problems

In operational planning problems, one looks at problems that deal with the day-to-day problems in a synchromodal network. This means that all these problems deal with uncertainty and stochasticity, which makes these problems complex. The decisions depend on the current information and an estimation of the future events. First, in Section 2.4.1 we discuss operational planning problems in general intermodal networks. In Section 2.4.2 we mention the papers that explicitly deal with synchromodality.

2.4.1 Operational planning problems in intermodal networks

One of the issues we have to look at is the reliability of a network. The planning of the commodities should be able to recover from disruptions. Most of the time disruptions cause re-planning. If this new planning is very different from the original planning, this might be costly. Three papers that look into disruptions and required recovery and preparedness actions are Huang, Hu and Zhang [66], Chen and Miller-Hooks [36], and Miller-Hooks, Zhang and Faturechi [88]. The paper by Huang, Hu and Zhang [66] looks at a new decision method for dealing with disruption events in intermodal freight transport. Chen and Miller-Hooks [36] define an indicator of network resilience that quantifies the ability of a network to recover from disruptions. They include a stochastic mixed-integer program for quantifying network resilience and identifying an optimal post event course of action. The paper by Miller-Hooks, Zhang and Faturechi [88] measures the networks maximum resilience level and determines the optimal preparedness and recovery actions as a two-stage stochastic program. A practical algorithm to deal with disruptions in railroad traffic is presented in the paper by Phillipson [103]. The two main questions that are answered are: when do we make a new planning and how do we create a new planning.

Another decision problem is that of resource management. Resource management deals with distribution of all resources through the network. Two issues in resource management are empty unit repositioning and the allocation and positioning of the operating fleet. These issues are discussed below.

- Empty unit repositioning problems arise when empty units need to be shipped from the location from which it was emptied to the location where it needs to be stocked again. Crainic et al. [41] discuss this problem. They describe the problem and give two formulations that provide a modelling framework. Erera, Morales and Savelbergh [50] look at the integration of the routing decisions with repositioning of the containers. They formulate the problem as a deterministic multi-commodity network flow over a time-expanded network. Di Francesco, Lai and Zuddas [46] address the repositioning of empty containers under port disruptions. They use a stochastic programming approach, in which different scenarios are included in a multi-scenario optimisation model. They show that only under normal circumstances the deterministic formulations are effective.

- One can also look at the allocation and positioning of the operating fleet, thus the vehicles that are used to transport the goods. In these problems there are a limited amount of vehicles, each with their own capacity. We want to optimally allocate the capacity of the fleet to the orders in order to maximise profit. Due to the high number of possibilities and integer constraints, simulation and approximation are mostly used to solve this problem. The papers by Topaloglu ([126] and [127]) and Topaloglu and Powell ([128], [129] and [130]) deal with fleet management and thus resource allocation. Bandeira, Becker and Borenstein [16] present a decision support system that integrates empty and full container positioning. The paper by Song and Dong [119] also look at the integration of both problems: cargo routing and empty container repositioning. They propose two solution methods for their optimisation problem.
Another problem in operational planning is the re-planning of the system, which involves the real-time optimisation of schedules, routes and the response to disruptions. This means that the system should continuously react and adapt in real-time. Literature about the real-time adaptations of intermodal systems is limited. The papers by Bock [23] and Goel [58] address the real-time issues into their models. Bock [23] proposes a real-time-oriented control approach. Vehicle breakdowns, traffic congestion and street blockages are integrated as possible disturbances. The paper also deals with dynamically incoming transportation requests. The author proposes a system where two plans are maintained: a relevant plan and a theoretical plan. The relevant plan is the one that is currently in execution and the theoretical plan is worked upon to see if future decisions need to be altered. An improvement heuristic is continuously applied. The paper by Goel [58] looks into the visibility of assets within an intermodal network. Here the current transportation plan can be adjusted to the known state of the system. The planning problem that is considered is the determination of shipments and their routes from suppliers to customers through an intermodal transportation network. The planning in this case is maintained with an update mechanism.

2.4.2 Synchromodality in operational planning problems

Zhang and Pel [137] develop a model that captures relevant dynamics in freight transportation. It consists of a demand generator (random sampling from historic data), an infra & service network processor (which generates the resource schedule), a schedule-based assignment module (which assigns the demand to resources) and a performance evaluator. The model can be used to compare intermodal and synchromodal transportation from different perspectives: economic, social and environmental. The authors use their model for a case study for the Rotterdam hinterland container transport and they show that synchromodality will likely improve service level, capacity utilisation and modal shift, but will not reduce delivery cost.

The paper by Mes and Iacob [87] searches for the $k$-shortest paths through an intermodal network. They present a synchromodal planning algorithm that takes into account time-windows, schedules for trains and barges and closing times of hubs and minimises costs, delays and CO$_2$ emissions. The $k$-shortest paths are then presented to a human planner, which can choose the best fitting path for an order by filtering the results. Their approach consists of offline steps and online steps. In the offline steps, the network is reduced by eliminating paths that are too far from the route. In the online steps an order is assigned to paths, by iterating over the number of main legs. A main leg in this paper is a certain train or barge. The assumption they make is that a cost efficient route consists of as few legs as possible. The online steps can be done after a disruption to make a new planning.

The paper by Rivera and Mes [101] looks at the problem of selecting services and transfers in a synchromodal network over a multi-period horizon. They take into account the fact that an order can be rerouted at any given moment. The orders become known gradually, but the planner has probabilistic knowledge about their arrival. The objective is to minimise expected costs over the entire horizon. They propose a Markov Decision Process model and a heuristic approach based on approximate dynamic programming.

2.5 Our research and its relation to existing literature

As we can see in the literature review above, a lot of literature focuses on the strategic and tactical planning problems. Literature on the operational and thus day-to-day planning of intermodal and thus synchromodal systems is limited. Here lie a lot of opportunities for further research. Certainly the use of stochasticity and uncertainty in synchromodal systems can be extended. Research here has mainly
focused on stochastic or uncertain demand, while there are a lot of factors within the system that are uncertain in real life. Research around the use of real-time data in the network is even more limited. Only Bock [23] and Goel [58] researched questions within an intermodal network, while keeping in mind the existence of this real-time data.

For our research we will look into operational planning problems and focus on the information that is available in the network. We analyse how shared information affects performance measures on certain intermodal networks. If all agents in the network have access to the information, they all can react on it. This means that in an agent-centric network, where each agent acts selfishly, this information may cause chaos. However, the available information can also be used to reach an optimal routing of the containers.

In the next chapter, Chapter 3, we discuss a framework for the synchronomal problems discussed in literature. This framework should help researchers to point towards solution methodologies that are common for certain similar problems. Except for the division of strategic, tactical and operational planning, synchronomal problems and models are most of the time not categorised. Therefore, it is difficult to see the similarities and differences in certain approaches. The framework categorises models in terms network characteristics, resource parameters and demand parameters. Chapter 4 and further discuss the problem of shared information in synchronomal networks.
Chapter 3

Framework of synchromodal transport planning problems

Authors: De Juncker, M.A.M., Huizing, D. and Ortega del Vecchyo, M.R.

In this co-authored chapter, we introduce a framework by which to classify mathematical models described in the literature on synchromodal transportation problems. The reason for such a framework will be elaborated on in the next section, but briefly put, this framework should help researchers and developers by pointing towards solution methodologies that are commonly used in their problem instance.

Section 3.1 will describe the context and motivation of the framework. Section 3.2 will introduce the classification framework and Section 3.3 will introduce a short-hand notation for it. In Section 3.4, some examples are provided. In Section 3.5, these examples are used to discuss strengths and weaknesses of the framework.

3.1 Motivation

In [102], seven critical success factors of synchromodality are discussed:

1. Network, collaboration and trust,
2. Awareness and mental shift,
3. Legal and political framework,
4. Pricing/cost/service,
5. ICT/ITS technologies,
6. Sophisticated planning,
7. Physical infrastructure.
From the list above, there are several different branches of knowledge involved in each of the critical factors (see Figure 3.1). Roughly, it can be argued that the first and second factor are mainly social problems, the third is a political problem, the fourth is a mathematical, social and political problem, the fifth is a technological problem, the sixth is a mathematical problem, and the seventh is a technological and constructional problem. In this work we are interested in those challenges within the realm of mathematics. These challenges are therefore directly related to the fourth and sixth success factors, but can also impact the first three in an indirect way, by promoting the effectiveness of synchromodality.

![Synchromodality Problems](image)

**Figure 3.1: Synchromodal Problems**

Synchromodal problems are often divided into three main categories: strategical, tactical, and operational. So is the case with mathematical synchromodal problems. These problems are related in a pyramidal-like structure (see Figure 3.2) in the following sense: tactical problems are usually considered where a specific strategical instance is given, and operational problems are frequently solved where a strategical and tactical structure is fixed. Although sometimes problems in two consecutive levels are solved simultaneously, for instance, in [18] the frequency of a resource is determined along with the flow of freight (that is, part of the schedules to resource and the freight to resources are solved at once).

Mathematical synchromodal transportation problems on a tactical or operational level are usually represented via tools from graph theory and optimisation [121]. However, more often than not, the similarities end there: most of the models used to analyse a synchromodal transportation network are targeted to a specific real problem of interest [121], and knowledge and methods of other branches such as statistics, stochastic processes, or systems and control are often used. The models emphasise on what is most important for the given circumstances. Consequently, mathematical synchromodal transportation problems on a tactical or operational level have been reviewed with approaches that may differ in many aspects:

- The exhaustiveness of the elements considered varies, e.g. traffic conditions are considered in some models (such as the one presented in [77]) but not all.
- The elements that can be manipulated and controlled may vary, e.g. the departure time of some transportation means may be altered if suitable (as it happens in the model of [18]) or it may be that all transportation schedules are fixed.
- The amount of information relevant to the behaviour of the network may vary, and if a lack of information is considered, the way to model this situation may also vary [101].
- Whether some other stakeholders with authority in the network are in the model, and if so, how their behaviour is modelled.

A model is not necessarily improved by making it increasingly exhaustive. As it happens with most model-making, accuracy comes with a trade-off, in this case, computational power. This computational
burden is an intrinsic property of operational synchromodal problems [112] and one that is of the utmost importance given the real-time nature of operational problems: new information is constantly fed and it should be processed in time.

There is no rule of thumb for making the decisions above. Each of the decisions mentioned above will shape the model, and likely steer its solution methods to a specific direction. Though literature reviews of synchromodal transportation exist [112, 121], it appears no generalised mathematical model for synchromodal transportation problems has been found yet, nor a way of categorising the existing literature by their modelling approaches. The framework for mathematical synchromodal transportation problems on a tactical or operational level we present in this chapter aims to capture the essential model-making decisions. This is done in an attempt to grasp the characteristics of the model/case in a compact way, enabling easy classification and comparison between models and cases, as well as a way to see the complexity of a specific case at a glance. Also, it provides a perspective to better relate new problems with previous ones, thus identifying used methodologies for the problem at hand.

### 3.2 Framework identifiers and elements

Within the framework we will refer to demand and resources. In synchromodal transportation models, demand will likely be containers that need to shipped from a certain origin to a destination. Resources can for example be trucks, train and barges. However, the framework allows for a broader interpretation of these terms. In repositioning problems, empty containers can be regarded as resources, whereas the demand items are bulks of cargo that need to be put in a container.

The framework has two main parts. The first part consists of the identifiers; these are specific questions one can answer about the model that depict the general structure of the model. The other is a list of elements; these elements are used to depict in more detail what the nature is of the different entities of the synchromodal transportation problem.
3.2.1 Identifiers

In this section we will elaborate on the identifiers of the framework. These identifiers are questions about the model. They identify the number of authorities, i.e. how many agents are in control of elements within the model. They will also identify the nature of different elements within the model. The list of elements will be discussed in detail in Section 3.2.2, but they are used to determine which components in the model are under control, which are fixed, which are dynamic and which are stochastic. For instance, the departure time of a barge may be a control element, but it could also be fixed upfront, or modelled as stochastic. Some of the questions address how the information is shared between different agents and if the optimisation objective is aimed at global optimisation or local optimisation. All the answers on these questions together present an overview of the model, which can then be easily interpreted by others or compared to models from the literature.

The identifiers that discuss the behaviour of the model in more detail are discussed below.

1. Are there other authorities (i.e. agents that make decisions)?
   Here it is identified if there is one global controller that steers all agents in the network or that there are multiple agents that make decisions on their own.
   - If there other authorities, how is their behaviour modelled: one turn only, equilibrium or isolated?
     If the previous question is answered with yes, i.e. there are multiple agents that make decisions, one needs to specify how these authorities react to each other. We distinguish three different ways for modelling the behaviour of multiple authorities in a synchromodal network:
     - One turn only: this means that each agent gets a turn to make a decision. After the decision is made, the agent will not switch again. For instance: we have three agents $A, B$ and $C$. Agent $A$ will first make a decision, then agent $B$ and then agent $C$. The modelling ends here, since agent $A$ will not differ from its first decision.
     - Equilibrium: the difference between “one turn only” and “equilibrium” is that after each agent has decided, agents can alter their decision with this new knowledge. In the same example: agents $A, B$ and $C$ make a decision, but agent $A$ then decides that due to the decision of agent $B$ to alter its decision. If nobody wants to alter their decision an equilibrium between the agents is reached.
     - Isolated: if the behaviour of the various authorities is isolated, it means that from the perspective of one of the authorities we only have limited information about the decisions of the other agents. For instance: agent $C$ needs to make a decision. It is not known what agents $A$ and $B$ have chosen or will choose, but agent $C$ knows historic data on the decisions of agents $A$ and $B$. Agent $C$ can then use this information to make an educated guess on the behaviour of agents $A$ and $B$.

2. Is information within the network global or local?
   This identifies if the information within the network is available globally or locally. If the information is locally available, it means that only the agents themselves know for example where they are or what their status is at a certain time. If the information is global, the network operator and/or all other agents know all this information as well.

3. Is the optimisation objective global or local?
   The same can hold for the optimisation objective. If all agents need to be individually optimised, the optimisation objective is local. If the optimisation objective is global, we want the best option for the entire network.
4. **Which elements do you control?**

Since we want to model a decision problem, at least one element of the system must be in control. For example: if one wants to model which containers will be transported by a certain mode in a synchromodal network, we have control over the demand-to-resource allocation. If we want to model which trains will depart on which time at certain locations, we have control of the resource departure time. An extensive list of elements is mentioned in Section 3.2.2.

Of course the controllable element can have constraints: for instance we can influence the departure times of trains, but they cannot depart before a certain time in the morning. This is still a controllable element. We thus consider an element a controllable element if a certain part of it can be controlled.

5. **What is the nature of the other elements** (fixed, dynamic, stochastic or irrelevant)?

The other elements within the network can also have a different nature. We distinguish four:

- **Fixed**: a fixed element does not change within the scope of the problem.
- **Dynamic**: a dynamic element might change over time or due to a change in the state of the system (e.g. the amount of containers changes the travel time), but this change is known or computable beforehand.
- **Stochastic**: a stochastic element is not necessarily known beforehand. For instance it is not known when orders will arrive, but it is a Poisson process. It might also occur that the time the order is placed is known, but the amount of containers for a certain order follows a normal distribution.
- **Irrelevant**: the list we propose in Section 3.2.2 is quite extensive. It might occur that for certain problems not all elements are taken into consideration to model the system. Then these elements are irrelevant.

6. **What is the optimisation objective?**

This identifier is for the optimisation objective. One can look at the exact same system but still want to minimise a different function. One could think of travel times and CO\textsubscript{2} emissions. It is also possible to provide a much more specific optimisation objective. Examples of optimisation objectives are given in Section 3.4.

### 3.2.2 Elements

In this section, we give a list of elements we believe exist in most synchromodal transportation problems. They are divided in two parts: resource elements and demand elements. The resource elements are all elements related to the resources, which are mostly barges, trains and trucks. However, for compactness we also view a terminal as a resource. The demand elements are all elements related to the demand, which are most of the time freight containers or empty containers. Most elements mentioned in this list are straight-forward, we mention small clarifications if necessary.

- **Resource elements:**
  - **Resource Type (RT)**: Different modalities can be modelled as different resource types. Also owned and subcontracted resources may be modelled as different resource types.
  - **Resource Features (RF)**: These features can be appointed to the different resource types or can have the same nature for the different types. For instance, it may be that there are barges and trains in the problem, but their schedules are both fixed, thus making the nature of the resource features fixed for both resource types.
• **Resource Origin (RO)**
• **Resource Destination (RD)**
• **Resource Capacity (RC):** Indication of how much demand the different resources can handle.
• **Resource Departure Time (RDT)**
• **Resource Travel Time (RTT):** Time it takes to travel from the origin to the destination (in the case of a moving resource).
• **Resource Price (RP):** This can be per barge/train/truck/... or per container.
  - **Terminal Handling time (TH):** Time it takes to handle the different types of modes at the terminal. This can again be per barge/train/truck/... or per container.

• **Demand elements:**
  - **Demand Type (DT):** One can also think of different types of demand. For instance, larger and smaller containers or bulk.
  - **Demand-to-Resource allocation (D2R):** The assignment of the demand to the resources.
  - **Demand Features (DF)**
    * **Demand Origin (DO)**
    * **Demand Destination (DD)**
    * **Demand Volume (DV):** It might be that different customers have a different amount of containers that is being transported. (Note that the demand element in this case will always be 1 container, since each container can have its own assignment.)
    * **Demand Release Date (DRD):** The release date is the date at which the container is available for transportation.
    * **Demand Due Date (DDD):** Latest date that the container should be at its destination.
    * **Demand Penalty (DP):** Costs that are incurred when the due date is not met or when the container is transported before the release date (this is sometimes possible with coordination with the customers).

### 3.3 Notation

In this section, we introduce some notation which makes it easier to quickly compare different models. Obviously, it is hard to keep a compact notation and still incorporate all aspects of a synchromodal system. Therefore, we made the notation as compact as possible and left out some of the details. When comparing models in detail it is easier to look at all answers to the identifiers mentioned in Section 3.2.1. Our notation has similarities to (among others), the framework of Kooiman for Time stamp Stochastic Assignment Problems [150], Kendall’s notation for classification of queue types [73] and the notation of theoretic scheduling problems proposed by Graham, Lawler, Lenstra and Rinnooy Kan [60].

A synchromodal transportation model is described by the notation:

\[ R|D|S \quad \text{or} \quad R|D|S|B, \]

depending on whether or not there are other authorities in the system. The letters denote the following things:

- **R:** resource elements,
- **D:** demand elements,
- **S:** system characteristics,
- **B:** behaviour of other authorities (if applicable).
Resource and demand elements

The first two entries in the notation can be filled with all elements mentioned in the list in Section 3.2.2. As mentioned before an element can be one of five different things: controlled, fixed, dynamic, stochastic or irrelevant. Let us elaborate on the notation of these differences. For the sake of example, we will use the element Demand-to-Resource allocation ($D2R$). We propose to use the following notation for the different natures of this element:

- controlled element: $[D2R]$,
- fixed element: $\overline{D2R}$,
- dynamic element: $\overline{D}\overline{2R}$,
- stochastic element: $\hat{D}\overline{2R}$,
- element irrelevant: $D\overline{2R}$.

Writing down all elements will still result in a large string of text. Therefore, we suggest to use $R$ and $D$ for the most common aspect and thus noting only the elements that are different. For example, for the resource elements everything is fixed, except for the departure time, which can be controlled. We would recommend the reader to write down: $\overline{R}, [RDT]$.

If different resource types or demand types have a difference in some of the elements one can also write this down. We propose to write down in between brackets the different resource type. For example: for barges everything is fixed, except for the dynamic capacity, but for the train the capacity is stochastic and the other elements are fixed. This can be noted in the following way: $\{\overline{R}, RC\}, \{\overline{R}, RC\}$. Note that this is the only way in which we incorporate the types into the notation. To know which types are used in the different models one has to look at the expanded notation. This choice is also made for the sake of compactness.

System characteristics

For the system characteristics we have developed a notation in which you have an answer to questions 1, 2 and 3 of the identifiers. Thus: are there other authorities, is the information global or local and is optimisation global or local.

The notation is based on Figure 3.3. In a similar way to this figure, the four options for the field System characteristics in the notation are:

- **selfish**: information global and optimisation local,
- **social**: information global and optimisation global,
- **cooperative**: information local and optimisation global,
- **limited**: information local and optimisation local.
In order to see if there are other authorities within the system we write down either an (1) or (1+) behind the option chosen. One could for example write down: social(1) or cooperative(1+).

**Behaviour of other authorities**

If there are other authorities within the system, their behaviour should be known (see question 1a in Section 3.2.1). The options are the same as discussed before:

- *one turn only*,
- *equilibrium*,
- *isolated*.

This field can be left blank if there are no other authorities in the system.

**Remarks**

The notation we developed does not include the optimisation objective. This is done on purpose. We would like to show all the model characteristics within the notation. Within a specific model there is of course an option to look at different optimisation objectives. Since these might be quite elaborate, we did not want to shorten these objectives to a few words. We think this will only result in unreadable notation. If one is interested, he/she can look at the entire list of identifiers.
This framework is developed in collaboration with multiple parties that study synchromodal systems. Therefore, we think we identified the resource and demand elements that are most common in synchromodal problems. However, for certain specific problems one might to extend the framework. We think this is easily done. For example, one can add some elements within the list of elements or a different nature of one of the elements. However, one must keep in mind that the scope of the framework mainly covers mathematical problems on the operational and tactical levels.

### 3.4 Examples

As discussed earlier, one of the ideas of the framework is that, when starting to work on a new problem, one can first classify the assumptions this model would need, then investigate papers that have similar classification. Therefore, we present a number of classification examples for both existing models as well as new problems. First, we answer the framework questions for the Kooiman pick-up case [76] in Table 3.2, and show how this can be written in our compressed notation. Afterwards, Table 3.3 shows compressed notation of some other problems described in papers, so that the interested reader can study more examples of our framework classification. Then, using Table 3.4, we examine some real-life cases and classify how we would choose to model these problems. These real-life problems do not yet have an explicitly described model, so this classification is based on how we would approach and model these practical problems, but other modellers may make other modelling decisions. Finally, the given examples will be used as input for discussion. We refer to table 3.1 for a reminder of the framework element abbreviations.

In the Kooiman pick-up case [76], a barge makes a round trip along terminals in a fixed schedule to pick up containers to bring back to the main terminal; however, the arrival times of the containers at the terminals are stochastic. At each terminal, a decision has to be made on how many containers to load onto the barge, and an estimate has to be made of how much capacity will be needed for later terminals, all while minimising the amount of late containers. The actual time of residing at the terminal is disregarded. We refer to table 3.2 for the answering of the framework questions.

<table>
<thead>
<tr>
<th>$RO$: resource origin</th>
<th>$DO$: demand origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$RD$: resource destination</td>
<td>$DD$: demand destination</td>
</tr>
<tr>
<td>$RC$: resource capacity</td>
<td>$DV$: demand volume</td>
</tr>
<tr>
<td>$RDT$: resource departure time</td>
<td>$DRD$: demand release date</td>
</tr>
<tr>
<td>$RTT$: resource travel time</td>
<td>$DDD$: demand due date</td>
</tr>
<tr>
<td>$RP$: resource price</td>
<td>$DP$: demand penalty</td>
</tr>
<tr>
<td>$TH$: terminal handling time</td>
<td>$D2R$: demand-to-resource assignment</td>
</tr>
</tbody>
</table>

Table 3.1: Abbreviations of the framework elements used in the compressed notation.
<table>
<thead>
<tr>
<th>Other authorities</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information global/local</td>
<td>Global</td>
</tr>
<tr>
<td>Optimisation global/local</td>
<td>Global</td>
</tr>
</tbody>
</table>
| Resource elements | $RT$: barges  
Controlled resource elements: none  
$RF$: fixed, except $\bar{HH}$ |
| Demand elements | $DT$: freight containers  
Controlled demand elements: $D2R$  
$DF$: fixed, except $\bar{DRD}$ |
| Optimisation objective | Maximise percentage of containers that travel by barge instead of truck |

Table 3.2: The framework questions applied to the Kooiman pick-up case.

Note that we have only taken barges into consideration as resources, not trucks. It would also have been possible to describe trucks as resources as well, but we have chosen to classify these as part of the lateness penalty, because there is no decision-making in how the trucks are used. Also, it may seem strange to speak of global or local information and optimisation when there are no other decision-making authorities. The information is considered global, because the only decision-making authority knows ‘everything’ that happens in the network; the optimisation is considered global, because the decision-maker wants to optimise the performance over all demand in the network put together, not over some individual piece or pieces of freight.

Using the framework notation, most of Table 3.2 can be summarised as follows:

$$\bar{R}, \bar{HH}\bar{D}, [\bar{D2R}], \bar{DRD}|social(1)$$

In Table 3.3, we apply the framework to more problems from academic papers. In this table, we include the optimisation objective to illustrate the wide range of optimisation possibilities. It is not actually necessary to describe the optimisation objective when using the compressed problem notation. In some cases, especially practical problem descriptions, optimisation objectives may not yet be explicitly known. Therefore, Table 3.4 leaves them out. In that table we review some practical problem descriptions and apply the framework to these descriptions.

39
Table 3.3: For selected papers, a classification of where their problem falls in the synchromodal framework.

<table>
<thead>
<tr>
<th>Paper (Reference)</th>
<th>Synchromodal Description</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behdani [18]</td>
<td>$R, [RDT]</td>
<td>D, [D2R]</td>
</tr>
<tr>
<td>Kooiman [76]</td>
<td>$R, TH</td>
<td>D, [D2R], D^RD</td>
</tr>
<tr>
<td>Le Li [78]</td>
<td>$R, RCT, RTT</td>
<td>D, [D2R], D^V, D^RD, D^DD</td>
</tr>
<tr>
<td>Lin [79]</td>
<td>$R, RP, RC</td>
<td>D</td>
</tr>
<tr>
<td>Mes [87]</td>
<td>$R, RC, RP, RCT</td>
<td>D, [D2R]</td>
</tr>
<tr>
<td>van Riessen [114]</td>
<td>${\tilde{R}, RO, RD, [RDT]} {\tilde{R}, RO, RD}, TH</td>
<td>D, [D2R], D^P</td>
</tr>
<tr>
<td>Rivera [101]</td>
<td>$R, \tilde{R}, D, [D2R]</td>
<td>social(1)</td>
</tr>
<tr>
<td>Theys [125]</td>
<td>$R, [RP, RCT]</td>
<td>D, [D2R], [DP], D^RD, D^DD</td>
</tr>
<tr>
<td>Zhang [137]</td>
<td>$R, \tilde{R}, D, [D2R]</td>
<td>social(1)</td>
</tr>
</tbody>
</table>

Table 3.4: For selected use cases, a classification of where a possible model for this problem would fall in the synchromodal framework.

<table>
<thead>
<tr>
<th>Use Case (Reference)</th>
<th>Synchromodal Description</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lean and Green Synchromodal [142]</td>
<td>$\tilde{R}, D, [D2R]</td>
<td>selfish(1)</td>
</tr>
<tr>
<td>Rotterdam – Moerdijk – Tilburg [143]</td>
<td>$R, \tilde{R}, RTT, TH</td>
<td>D, [D2R]</td>
</tr>
<tr>
<td>Synchromodaily [146]</td>
<td>$R, [RDT]</td>
<td>D, [D2R]</td>
</tr>
<tr>
<td>Synchromodal Control Tower [144]</td>
<td>$R, [RC], RP, RTT, TH</td>
<td>D, [D2R], [DP]</td>
</tr>
<tr>
<td>Synchromodal Cool Port control [145]</td>
<td>$R, [RDT], \tilde{RTT}</td>
<td>D, [D2R], \tilde{D^DD}, D^P</td>
</tr>
</tbody>
</table>

Another example we reviewed is the modelling of an agent-centric synchromodal network. Here all agents want to be at their destination as fast as possible, but everyone does share the information about where they are and where they are going with everybody else in the network. Table 3.5 shows the answer to...
the questions of the framework. In the short notation this problem is:
\[\mathcal{R} | \hat{D}, [D2R], \mathcal{D}\mathcal{P} | selfish(1+) | equilibrium\]

<table>
<thead>
<tr>
<th>Other authorities</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information global/local</td>
<td>Global</td>
</tr>
<tr>
<td>Optimisation global/local</td>
<td>Local</td>
</tr>
</tbody>
</table>
| Resource elements | \(RT\): barges, trains and trucks  
\(RF\): fixed |
| Demand elements | \(DT\): containers  
\(DF\): stochastic, except \(\mathcal{D}\mathcal{P}\) |
| Optimisation objective | Minimise travel times |

Table 3.5: The framework questions answered for the agent-centric synchromodal network.

3.5 Discussion

The examples from the previous section show some strengths and limitations of the classification framework, which are discussed in this section.

One of the goals of this framework was to offer guidance when tackling a new problem: as an example, if the problem from the Synchromodaily [146] case is modelled in a non-stochastic way, we can now see that it may be worthwhile to study the solution method presented by Behdani [18], because they then have the same compressed framework classification. If such a record is kept of papers and models, this could greatly improve the efficiency of developments in synchromodal transport. This would fulfil the second goal of the framework: to collect literature on synchromodal transportation within a meaningful order.

The final goal of this framework was to expose and compare relationships between seemingly different problems: for example, we can now see that the problems described by Le Li [78] and Theys [125] have similarities, in that they investigate negotiation between parties and do not focus on timeliness of deliveries. Similarly, we can see that the model assumption Mes [87] makes in disregarding resource capacity, is an unusual decision.

In the Synchromodaily case [146], our interpretation of the problem implies that the demand features are stochastic. However, the problem could also be approached in a deterministic way, depending on choices that the modeller and contractor make based on the scope of the problem, the requirements on the solution and the available information. This shows the most important limitation of the classification framework: what classification to assign to a problem or model remains dependent on modelling choices, as well as interpretation of problem descriptions. Even without framework, however, modelling choices will always introduce subjective elements into how a real-world problem is solved. This framework can be used to consistently communicate these underlying model assumptions.

A second limitation of the framework is that, because of the large amount of elements described in it, two similar problems are relatively unlikely to fall in the exact same space in the framework because of their minor differences. Therefore, one should not only look for problems with the exact same classification, but also problems with a classification that is only slightly different. In a more general sense, solution
methods may apply to far more than one of these very specific framework classes. If two problems have the exact same controlled elements, it is imaginable that their models and solution methodologies may largely apply to the other. As a point of future research, it could be interesting to investigate which classification similarities are likely to imply solution similarities, which may also be a stepping stone towards a general solution methodology.

As a final limitation, the compressed notation does not reveal that the paper by Lin [79] and the ‘Synchromodal Cool Port control’ [145] case both focus on perishable goods. This shared focus is not only cosmetic: mathematically, it may imply objective functions and constraints not focused on in other cases. To mitigate this limitation, we advise anyone using the framework to offer both a compressed and an extended description of their problem or model.
Part III

Optimal routing of containers in synchromodal transportation networks
Chapter 4

Problem description

In the previous part of this thesis, we described synchromodal transportation and its current research. Here we focused mainly on planning problems and described that research in operational, day-to-day planning problems is scarce. Our research, which is described in the next part, focuses on day-to-day planning (under uncertainty) of containers in a synchromodal system, i.e. a system that includes shared information for all agents.

4.1 Description of the research

As described in Chapter 1, our main focus is to investigate the effects of information availability in agent-centric systems. We are interested to see what global information can do for the optimal routing of containers through an intermodal network. Gaining information would likely mean that agents are more adapt at finding their optimal route. However, it is also possible that congestion increases, increasing costs for all agents involved.

We distinguish two types of available information: public information and private information. Public information is information that is available for everyone. This means that for public information we do not need different stakeholders to cooperate. The public information we encounter in our problem is information about the occupancy of different links. This means that we know, for any point in time, how many agents are on every link of the network. This public information is currently used for road networks. For example: route guiding systems already have an option to recalculate your shortest path based on information about congestion in the network. Private information on the other hand, does require different stakeholders to cooperate. The private information we encounter is information about upcoming orders. Different logistic service providers normally know what orders are going to arrive in the near future. If these logistic service providers would be willing to share this information with all stakeholders, all stakeholders can react upon this information. This means that agents also have information about the future occupancy of certain links.

This project investigates the effects on the costs and thus on the optimisation objectives of all agents with different information availability. This means that we assume cases where only public information is available and cases where agents have access to both public and private information. We develop five different models for the analysis. We will also investigate the similarities and differences between these models.
4.1.1 Research questions

Our problem can be summarised in the following main research question:

- **What is the optimal routing for containers in an agent-centric synchromodal system?**

The synchromodal aspect in this research question is the use of shared information. All agents have access to the information that is available in the network. The research question will be divided into smaller research questions, since the sort of information might differ. We encounter public and private information as described before. These different types of information result in three sub-questions, mentioned below:

- **What is the optimal routing for containers in an agent-centric network with only public information?**
- **What is the optimal routing for containers in an agent-centric network with full information, both public and private?**
- **What is the optimal routing for containers in an agent-centric network with stochastic full information?**

The last question mentions stochastic information. Here we assume that the private information of incoming orders is shared for all agents, but is stochastic in nature. The information of incoming orders might not be accurate, but based upon estimates.

For the analysis of an agent-centric synchromodal network we develop five different models and thus five different solution methods. Two of these models assume only public information, the others assume both public and private information in various degrees of stochasticity. These different models immediately raise one other research question:

- **What are the similarities and differences between the developed solution methods?**

4.2 Approach

As mentioned in the literature review in Chapter 2, research on the operational planning aspect in synchromodal systems is rare. Therefore, it is difficult to search for solution methods on the optimal routing of containers. However, a research area with a lot of similarities to our problem is the area of *Dynamic Traffic Assignment* (DTA). In a traffic system all cars also act as individual agents that want to optimise their own objective. DTA models aim to find an optimal assignment of traffic on certain roads, i.e. they want to find the routes of all vehicles present in the system that minimise an objective function. These models aim to capture all the interactions, randomness and time-dependency within a traffic system, making them a perfect starting point for our research. Mathematically, it does not really matter if we want to assign vehicles to roads or containers to links in an intermodal network.

We start our research by looking into the literature on these DTA models. This literature review is found in Chapter 5. Here we also mention how certain models can be used in our research. Further down the report we explain our models. In Chapter 6 an analytic model that starts by looking at a road network and is extended into a synchromodal network. Chapter 7 describes our simulation-based solution model.
• **Analytic model:** the first model is an analytic model based on stochastic networks. We modelled the agent-centric network as a *Multi-Class Open Product-Form Network*. Here each link in the network is represented with different queuing models, that aim to capture the relevant processes of containers moving over a road, rail, or waterway.

• **Simulation-based model:** the other model is an event driven simulation, that simulates the agents going through the synchromodal network.

For the simulation-based model we developed four different methods to generate routes for the containers. All methods aim to minimise certain costs. Two of them are heuristics and one is an extension on a DTA model found in Chapter 5, which is adapted to include different transportation modes. The four methods are described below:

• **Heuristic 1:** The first heuristic is a naive implementation with public information. We assume that each agent checks the public information at departure and will react accordingly. This means they will take the shortest path for the current state of the network. After the choice is made, they will not deviate.

• **Heuristic 2:** The second heuristic is very similar to the first, except the fact that agents do switch routes before reaching their destination. This means that at each decision point, i.e. an intermediate node in the network, they again check the state of the network and reroute if necessary.

• **Simulation-based solution method:** the simulation-based solution method assumes full information. This means that this algorithm assumes that each agent knows exactly what is going to happen for a certain planning horizon. This includes public as well as private information. With this information, the algorithm seeks to find the optimal routes for all agents that arrive somewhere in the planning horizon.

• **Simulation-based solution method in a rolling-horizon framework:** Private information will not always be accurate. Therefore, we also developed a model that assumes that for a short time period reliable estimates are known and for a longer time period estimates are known that are less reliable. This algorithm relies heavily on the simulation-based solution method.

These different methods will then be applied to simulated instances of orders on an intermodal network. Unfortunately, real data is not yet available in this project and therefore we discuss part of the Trans-European Transport Network with parameters that we will choose. With these parameters we aim to simulate a realistic instance of a synchromodal system. The network and its parameters can be found in Chapter 8.
Chapter 5

Review of Dynamic Traffic Assignment and information availability in traffic systems

In the previous chapter we mentioned that we will use Dynamic Traffic Assignment (DTA) models as a starting point for our solution methods. This chapter describes the literature and current practice in traffic modelling. In Section 5.1 we list some important literature on DTA. Section 5.2 describes research and current practice on information availability in traffic systems. Lastly, in Section 5.3 we discuss which of the current models can be extended to a synchromodal transportation model.

5.1 Dynamic Traffic Assignment

In our agent-centric synchromodal network, the objective of the agents is to minimise their individual objective functions. This means they have to decide on their route through this synchromodal network. In a traffic situation, agents can also decide on their route through the road network. This means one looks at the assignment of traffic to certain roads, therefore this problem is called Traffic Assignment in the literature.

Research in traffic assignment tries to create and implement a model which makes it possible for vehicles to reach their destination in the shortest possible time or with minimum costs. Modelling choices determine whether the road network that is investigated is static or dynamic. When a network is static all capacities and vehicle flows from and to certain nodes do not change over time. In the dynamic case the road capacity, vehicle flow and maybe other elements are time- and state-dependent. The static case is fairly easy: given a flow of vehicles and their origin-destination pair the shortest path can be easily calculated by a well-known technique like Dijkstra’s algorithm. In the static case the amount of time needed to travel a road does not change. Obviously, this is not the case in real life and therefore we focus on the dynamic case; Dynamic Traffic Assignment.
In Dynamic Traffic Assignment (DTA) problems one can take one of two approaches in choosing objectives:

1. **System optimal (SO):** in this case one wants to optimise a system objective; e.g. congestion or average travel time. This also means that all vehicles are controlled by a central controller.

2. **User equilibrium (UE):** here every vehicle in the system wants to optimise an individual performance measure; e.g. travel time or costs.

Our agent-centric synchromodal system obviously falls under the user equilibrium approach, since all agents want to minimise their own objective function. However, it is still interesting to see how the system optimum is modelled, since this can be used as a reference point for other performance indicators like congestion.

There are a few models developed for these DTA problems, but there is no model that provides a universal solution for general networks [99]. In DTA models, there is a trade-off between traffic realism and the theoretical guarantee of properties such as existence, uniqueness and stability.

Peeta and Ziliaskopoulos [99] give a nice overview of the different modelling approaches to DTA problems. These approaches can be divided into four categories: *mathematical programming*, *optimal control*, *variational inequality* and *simulation-based*. Most model formulations focus on the system optimal and user equilibrium objectives.

All analytic approaches, i.e. mathematical programming, optimal control and variational inequality, struggle with two main requirements of traffic realism: a first-in, first out (FIFO) requirement and the artificial holding back of traffic.

- **FIFO requirement:** In a real-life traffic network there will always be overtaking on an individual level. However, this FIFO requirement prohibits some commodity, e.g. an entire flow of vehicles from one origin, to overtake another commodity to reduce costs. In our intermodal network this would mean that for example a barge leaving the port later but with more containers will arrive before the previous barge travelling the same waterway. This is of course not realistic.

- **Artificial holding back of traffic:** If the formulation does not specify how long certain traffic may be held back, it might be beneficial to hold back a certain flow for hours, which is not reasonable in real life. For example, the road network has a certain junction with two main flows: flow 1 and flow 2. If flow 1 is very large, it might be beneficial to keep the junction “open” (think about green traffic lights) for flow 1 and “closed” (red traffic lights) for flow 2 for an entire day. However, in real life no vehicle would wait till nightfall to cross a junction. All analytic approaches have no satisfactory solution to these requirements. This is the reason for the existence of simulation-based models, which can handle more realistic instances.

We mention interesting literature on all four approaches. We also describe the advantages and disadvantages of using certain approaches.

- **Mathematical programming formulations.** Mathematical programming DTA models aim to formulate the problem as a mathematical program. Time in these models is discretized. The first formulation was by Merchant and Nemhauser [86], which formulated the deterministic, fixed-demand,
single-destination, single-commodity, SO case. Birge and Ho [22] extend this model to the stochastic case by allowing for random demand desires. Jansen [68] describes the UE DTA problem as a mathematical program. However, all formulations are non-convex because of the FIFO requirement. While there is enough literature on non-convex optimisation, in a DTA context analytical and computational tractability are lost for general networks. Together with the difficulty to prohibit holding-back of traffic, mathematical programming formulations lack efficient solutions for realistic instances. Carey and Subrahmanian [30] illustrate some of the issues that arise because of the FIFO requirement and the holding-back of traffic in mathematical programming formulations.

- **Optimal control formulations.** Optimal control theory DTA formulations are in continuous time, unlike the mathematical programming formulations. Here the origin-destination rates are assumed to be known continuous functions of time. For the formulations we refer to [57], [111] and [110]. The main issue with optimal control formulations is that there is no efficient solution algorithm. As mentioned before, these formulations also have no explicit constraints for the FIFO requirement and the holding back of traffic. Therefore, new research has been done in variational inequality formulations.

- **Variational inequality formulations.** Dafermos [45] introduced the variational inequality approach in static traffic assignment. Variational inequality formulations are used in equilibrium problems. One defines an inequality involving a functional, which has to be solved for all possible values of a variable. One standard example problem is the problem of finding the minimal value of a differentiable function \( f \) in a closed interval \([a, b]\). If \( x \in [a, b] \) is the point where this minimum is, then there are three cases:
  1. \( a < x < b \), thus \( f'(x) = 0 \)
  2. \( x = a \), thus \( f'(x) \geq 0 \)
  3. \( x = b \), thus \( f'(x) \leq 0 \)

  One can summarise this problem as the problem of finding \( x \in [a, b] \) such that \( f'(x)(y - x) \geq 0 \) for all \( y \in [a, b] \).

  Nagurney [91] provides a summary of variational inequality formulations and addresses various equilibrium problems in network economics, under which the traffic network equilibrium. Variational inequality formulations can handle more realistic traffic scenarios, but the approaches are computationally intensive. Also, the problems with the FIFO requirement and the holding-back of traffic remain.

- **Simulation-based models.** The simulation-based DTA models use a traffic simulator in order to handle realistic traffic scenarios. The main issue with simulation-based models is that theoretical insights cannot be gathered from the models.

  The solution methods in these simulation-based models often use the traffic simulator as part of the solution. This is called the predictive-iterative method, where the simulator is used in each iteration to predict future traffic conditions given a certain route assignment. Based on these predictions a new route assignment is determined and so on. One of these iterative models is described by Peeta and Mahmassani [98]. Here they describe a solution algorithm for SO and UE DTA that uses the traffic simulator DYNASMART. A similar iterative approach for a UE DTA is taken by Kaufman, Smith and Wunderlich [72]. These models are much more realistic than the analytic ones and therefore widely used in analyses. However, deployment in real life is only feasible if the algorithms are computational efficient. Peeta and Mahmassani [100] develop rolling horizon DTA models that use their solution algorithm from [98] for real time deployment. Similarly, Ben-Akiva et al. propose DynaMIT in [21] (and its route guidance in [20]), which uses a demand and supply simulator to generate UE route guidance under a rolling horizon framework.

As described above the only realistic models are the simulation-based models. Therefore, these are used more often to investigate network performance on real life instances. The algorithm described by Peeta
and Mahmassani in [98] is an algorithm that searches for the optimal routing in SO and UE DTA and includes research on information availability. This algorithm for the user equilibrium case is an interesting starting point for our solution algorithm, since it is easily extended into synchromodal systems.

5.2 Information availability in traffic systems

We are interested in how certain information affects synchromodal systems. Therefore, we are also interested in the research and current practice of information supply in road networks. This section elaborates on some of the research and on the current practice of supplying information to traffic users throughout The Netherlands.

**Research of real-time information in the literature.** The paper by Mahmassani and Jayakrishnan [82] describes a modelling framework to analyse the effect of in-vehicle real-time information. The framework consists of a simulation component and a user decisions component. The simulation component simulates the traffic flow through the network and the user decisions component determines users’ responses to the information. The paper uses a simulation approach in order to deal with the complexity of the interactions. The framework can be used in order to look at the effect of the fraction of users that is equipped with in-vehicle navigation systems on overall system performance. The paper describes results on various percentages of people who have access to the real-time information and their indifference towards this information, i.e. how likely are people to actually switch route.

Dia [47] studied an agent-centric network with real-time information in his paper. Here the model also consists of a traffic simulation component and a decision component. The decision component determines drivers’ responses to the information given in a more complex way than in other literature. The paper uses results from a behavioural survey to determine the factors that influence route choice decisions. Each driver is programmed as an Intelligent Agent. An Intelligent Agent can be thought of as computer surrogates for a person that fulfil a stated need or action. They comprise knowledge about the needs, preferences and patterns of behaviour of that person.

**Information supply in traffic systems throughout The Netherlands.** The site http://www.traffic-quest.nl (accessed on 06-03-2017) is a site from a collaboration between Rijkswaterstaat (Ministry of Infrastructure and the Environment) and some companies in The Netherlands. It provides information about the current state of the art methods used in The Netherlands on traffic issues. They also describe an investigation in traffic information [147]. Traffic information is considered: services that give real-time traffic information or traffic guidance to travellers about the conditions on the road, delays caused by accidents, accessibility of parking spaces and road maintenance. The need for information is biggest in situations that are uncertain.

The current ways of displaying this information to road users in The Netherlands are:

- Matrix signs: show speed limit, but also give information about traffic jams.
- DRIP (Dynamic Route Information Panels): programmable signs that inform about travel times or accidents/road maintenance.
- GRIP (Graphical Route Information Panels): graphically show the circumstances on the roads.
One can see that it is easy to have access to the information about congestion of Dutch roads. In this research we assume that it will only take time before everyone in the network has access to the public information at every given time.

5.3 Research used for expansion in synchromodal systems

Our research focuses on the real life implications of synchromodal transportation. This means we need our solutions to be able to handle realistic instances. Therefore, the literature on the analytic approaches of dynamic traffic assignment might not be good enough to start with. We develop another analytic approach based upon a stochastic network. We hope that this new approach is better in analytically gaining insights in network performance measures like sojourn and waiting time. However, as seen in the literature on traffic networks, the analytic approach might lack realism and therefore we also develop simulation-based models.

The stochastic network we build is a Multi-Class Open Product-Form Network and described in Chapter 6. For this model, we first focus on a road network and describe the parameters of the queues in this network. From here, we add links in the network to model trains and barges as in a synchromodal network and also discuss the parameters for these links.

The simulation-based approach is based on the approach by Peeta and Mahmassani [100] for a traffic network. We modify the solution procedure to accommodate for the trains and barges from a synchromodal network. The details of this approach are described in Chapter 7.
Chapter 6

Multi-Class Open Product-Form Networks

As discussed in the previous chapters we look at stochastic networks to analytically analyse our intermodal network. As there are no specific models for traffic networks, we start with a model for a traffic network. This model is a Multi-Class Open Product-Form Network. In this chapter we discuss the theory about multi-class open product-form networks in Section 6.1. Then in Section 6.2 we explain how we modelled the traffic network as a multi-class open product-form network. In the last section, Section 6.3, we expand the traffic network model into a model that includes trains and barges to make a model for an intermodal network.

Note that the routing probabilities in the road network are dynamic. An agent can switch routes depending on the state of the system. However, literature about stochastic networks with dynamic routing is limited. One of the few papers on this topic is by Suhov and Vvedenskaya [123]. They discuss a queuing network where each station consists of $N$ single exponential servers of rate one. A task arrives and chooses at random a certain amount of servers in each station and then joins the shortest queue from the sample. The authors of the paper discuss the capacity domain and properties of the limit of this model as $N \rightarrow \infty$. This means that the only results of queuing networks with dynamic routing are on networks where the number of servers tend to infinity. This is obviously not the case in our route network. Therefore, we will focus the modelling on queuing networks with static routing. In a network with static routing, the agents cannot react on information about the network. The dynamic routing and the use of information will be evaluated with the simulation-based models, see Chapter 7.

However, in this stochastic network performance indicators like sojourn time and waiting time can be easily computed. For a first approximation one can see how certain parameters can affect the waiting time of agents in the network based on the model described in this chapter. Besides being a bench mark for practical cases, we can also see this analytic network as a reason that we resort to simulation-based models. As discussed earlier other analytic models lack the realism we require in our solution method and thus cannot be used. This analytic approach does better in the FIFO-requirement and the artificial holding back of traffic, but knows other limitations. Therefore, we developed simulation-based models too, which are described in Chapter 7.
6.1 Theory of Multi-Class Open Product-Form Networks

In this section we mention the important parameters and aspects of a multi-class open product-form network. Most information for this section is gathered from lecture notes [149].

A multi-class open product-form network is a stochastic network. A stochastic network consists of $M$ queues and customers can go from a certain queue $i$ to another queue $j$ with a certain probability. Note that we include the customer(s) in service when we refer to the queue, therefore not all customers have to be physically in a queue. We will be modelling an open network, which means that customers arrive from outside the network and eventually leave the system. Since not all customers are statistically identical, we introduce multiple classes in the network. The parameters in such a network are mentioned below.

- $M$: number of queues
- $K$: number of different classes of customers
- $Q_i$: notation for queue $i$
- $n_{ik}$: number of class-$k$ customers present at queue $i$
- $n_i = \sum_{k=1}^{K} n_{ik}$: total number of customers present at queue $i$

Arrivals Arrivals from outside can occur at each queue and are assumed to follow a Poisson process. This means that the inter-arrival times follow an exponential distribution.

- $\lambda_k$: parameter of Poisson arrival process of customer class $k$

Routing probabilities We need to know how the customers move through the network. Therefore, we need routing probabilities. We are modelling an open network. This means that we have an environment outside the network from which customers arrive and to which they will return after they have been served by the network. We will denote this outside environment with a dummy queue, $Q_0$. The probability that their first visit is to $Q_i$ is $p_{0i}(k), i = 1, \ldots, M, k = 1, \ldots, K$. Thus the routing probabilities are:

- $p_{0i}(k)$: probability for a customer of class $k$ to arrive at $Q_i$.
- $p_{ij}(k)$: probability for a customer of class $k$ to move from $Q_i$ to queue $Q_j$
- $p_{i0}(k) = 1 - \sum_{j=1}^{M} p_{ij}(k)$: probability for a customer of class $k$ to leave the system from $Q_i$

The probability to leave the system from $Q_i$, $p_{i0}(k)$, should for all classes be strictly positive for at least one queue. Otherwise, it would not be possible to leave the system and the system would be unstable. Note that this condition is not sufficient for stability of the entire system, we will mention another condition for stability further down.
**Service**  For the service we distinguish three items: service time distribution, service discipline and a state-dependent service rate. The state-dependent service rate can be used to model the different types of queues. For example \( r_i(n_i) = 1 \) for all \( i \) means that it is a single unit-rate server. \( r_i(n_i) = \max\{n_i, C_i\} \) denotes that \( Q_i \) has \( C_i \) identical servers. Another example is \( r_i(n_i) = n_i \), which means that \( Q_i \) has infinitely many servers. In Section 6.2 we will see how this service rate is used for our traffic model.

The parameters that deal with the service of a customer are:

- \( B_i(k) \): distribution of service time at \( Q_i \) for customers of class \( k \) with mean \( \beta_i(k) \).
- The service discipline, e.g. First Come First Served or Processor Sharing.
- \( r_i(n_i) \): state-dependent service rate for \( Q_i \) when \( n_i \) customers are present.

For this type of network we want to derive the stationary distribution of the queue-lengths. This stationary distribution has a nice product-form (see below) under certain conditions. These conditions have to do with the service disciplines and service time distributions. If these conditions are not met, a product-from for the joint queue length distribution is not known.

In general we can distinguish two cases for which the product-form will still hold:

1. Customers belong to different classes, but still all have identically and exponentially distributed service times and the service discipline is First Come First Serve (FCFS).
2. Customers have class-dependent and generally distributed service times and the service discipline is symmetric, which we will describe below.

A service discipline is symmetric if there exist probability vectors \( \phi(n) := (\phi_1(n), \ldots, \phi_n(n)) \) such that the following two things hold:

1. If there are \( n \) customers present at the queue, the customer in position \( m \) of the queue receives \( \phi_m(n) \) of the total service rate \( r(n) \).
2. If there are \( n - 1 \) customers present at the queue and a new customer arrives, it joins the queue in position \( m \) with probability \( \phi_m(n) \). This means that all customers behind this customer, i.e. in positions \( m, \ldots, n - 1 \) move backward one position.

Some examples of symmetric service disciplines are listed to clarify the probability vector \( \phi(n) \):

- **Last-Come First-Served Preemptive Resume (LCFS-PR).** In this service discipline the customer that arrives at the queue will be served immediately. This means that the service of the customer in service, if any, is stopped and continued after a customer is finished at the server. (Note that this might be even after some other services, since customers can keep arriving at the queue.) In this service discipline one can take \( \phi(n) = (1, 0, \ldots, 0) \), i.e. \( \phi_1(n) = 1 \) and \( \phi_m(n) = 0 \) for \( m = 2, \ldots, n \).
• **Processor Sharing (PS).** In a Processor Sharing service discipline each customer in the queue will get an equal share of the service rate. This means that with a service rate of 1 and \( n \) customers in the queue, every customer gets an equal share of \( \frac{1}{n} \).

One should take \( \phi_m(n) = \frac{1}{m} \) for all \( m = 1, \ldots, n \).

Note that this service discipline can also be used to model an infinite-server queue. In this case \( r(n) = n \) and thus the fraction of the total service rate each customer receives is: \( \frac{1}{n} \cdot n = 1 \).

In conclusion, if each queue of the stochastic network is either of the two cases mentioned above, the product-form will hold. This means that not all queues have to fall in the same category. It is possible that several queues are FCFS with an exponential distributed service time and others have a general service time distribution, but the service discipline is symmetric.

**Traffic equations** If we denote by \( \gamma_i(k) \) the total arrival rate of class-\( k \) customers at queue \( i \), we know that it satisfies the traffic equations, see Equation (6.1), when the system is stable.

\[
\gamma_i(k) = \lambda_k p_0(k) + \sum_{j=1}^{M} \gamma_j(k) p_{ji}(k) \quad i = 1, \ldots, M
\]  

(6.1)

**Stability** Denote by \( \rho_i(k) \) the load of customer class \( k \) at queue \( i \), \( \rho_i(k) := \gamma_i(k) \beta_i(k) \). The system is stable if the condition in Equation (6.2) holds.

\[
\rho_i < r_i^* \quad \forall i = 1, \ldots, M
\]

where

\[
r_i^* := \lim_{n_i \to \infty} \inf_{n_i} r_i(n_i)
\]

\[
\rho_i := \sum_{k=1}^{K} \rho_i(k)
\]

(6.2)

**Stationary probabilities** The state of the system can be described by a vector:

\[
N(t) = (N_{11}(t), \ldots, N_{1K}(t), N_{21}(t), \ldots, N_{2K}(t), \ldots, N_{ik}(t), \ldots, N_{MK}(t)),
\]

where \( N_{ik}(t) \) denotes the total number of class-\( k \) customers present at queue \( i \) at time \( t \). The stationary probability that the system is in state \( n \in \mathbb{N}^{KM} \) is denoted by \( \pi(n) := \lim_{t \to \infty} \mathbb{P}(N(t) = n) \). This stationary probability has a product form as shown in Equation (6.3).

\[
\pi(n) = \prod_{i=1}^{M} \phi_i(n_i)
\]

with

\[
n_i = (n_{i1}, \ldots, n_{iK})
\]

\[
\phi_i(n_i) = G_i^{-1} \frac{1}{\prod_{m=1}^{n_{i1} + \cdots + n_{iK}} r_i(m)} \binom{n_{i1} + \cdots + n_{iK}}{n_{i1} \ldots n_{iK}} \rho_i(1)^{n_{i1}} \cdots \rho_i(K)^{n_{iK}}
\]

\[
G_i = \sum_{m_i=0}^{\infty} \frac{\rho_i^m}{\prod_{m=1}^{m_i} r_i(m)}
\]

(6.3)
Remarks  \( \phi_i(n_i) \) as described above, is equal to the joint stationary distribution of the numbers of customers at queue \( i \). This means that this distribution is the same when a queue operates in isolation as well as in a network. Secondly, the numbers of customers present at the various queues are independent in equilibrium. Note that this is only true in equilibrium, the numbers of customers at the various queues are of course dependent throughout time, because of the routing probabilities.

6.2 Modelling of a traffic network

In this section we elaborate on how we can model a traffic network as a multi-class open stochastic network. Thus, how do we go from having a network of cities and roads to a model with queues and routing probabilities. We use the network shown in Figure 6.1 as an example throughout this section.

![Figure 6.1: Example of a road network with 3 cities and 2 villages.](image)

We have to think about the concepts of the queuing systems described earlier applied to the traffic network. An important observation is that in a queuing system the server and waiting area are at a node (at the server), while in the road network the queues accumulate on the edges (roads) between the nodes (cities). In order to obtain a stochastic network we thus have to interchange the roles of edges and nodes. This means that the servers of our road network are actually the roads themselves.

Our service time distribution and service discipline are chosen according to the two cases we mentioned in the previous section. As mentioned there, service times in a product-form network can either be: exponentially distributed with a FCFS service discipline or generic distributed with a symmetric service discipline. In a road network it makes sense to model the service discipline as First Come First Served. Namely, a car that is earlier to arrive at a certain road will also be served earlier than a car that arrives later. However, if we would model the roads with FCFS, we would need the service time distributions to be exponential to keep the product-form. An exponential service time would mean that there is a non-zero probability for the cars to traverse the road in almost no time. For modelling the road network we therefore choose to introduce two independent queuing systems per road.

- The first system is a system with infinite servers and a deterministic service time. We mentioned before that an infinite service queue is a special case of a processor sharing service discipline, where \( \phi_m(n) = \frac{1}{n} \) for all \( m = 1, \ldots, n \) and \( r(n) = n \). Choosing to model this queue as an infinite server queue, will leave the possibility for the service time distribution to be chosen arbitrarily. We
choose a deterministic service time for this distribution. The deterministic service time represents the length of the road. In other words, the first queuing system of a road, models the time it takes just to traverse the road without any traffic.

- A second queuing system is introduced that models the traffic on the road. Here we work with a single server queue, so \( r(n) = 1 \) for all \( n \). The service discipline is again a processor sharing service discipline where \( \phi_m(n) = \frac{1}{n} \) for all \( m = 1, \ldots, n \). Since this queue only has a single server, we see that the service rate for everybody in the queue is dependent on how many people are on the road. This is the same when a traffic jam occurs, everybody will still be able to use the road, but the speed at which they are served is lower. The service time distribution can again be chosen arbitrarily. Whichever distribution we chose here has less effect on the total modelling of this queue, therefore we chose an exponential distribution. An exponential distribution is widely used as the distribution of a service time.

One can see that for both queues we have a symmetric service discipline, ensuring the product-form for the stationary distribution to hold.

Introducing the two queues per road means that the traffic network will be mapped to a stochastic network that has twice as many queues as roads in the original network. For compactness we will identify every first system, i.e. the system with deterministic service times, with server 1 of a road. Every second system, i.e. the system with exponential service times, will be identified as server 2 of a road. In Figure 6.2 one can see the two queues we mentioned for the roads in our example. We only see the servers in the figure, since both queues have no actual waiting queue.

![Figure 6.2: Two queues per road in the model of the road network.](image)

Now we have to think about all the details of our model. We discuss these in the same order as we did in Section 6.1. The exact parameters as arrival rate, service distribution and routing probabilities will be discussed in Chapter 8. In that chapter, we analyse one network with all the models that we developed.
and also discuss the parameters of that network. The current chapter is a more general view on road and intermodal networks as stochastic networks.

Firstly, we have to denote what the classes are in our model. All agents are considered equal in our model except for their origin and destination. Therefore, we define the classes according to the origin-destination pair of a car. We assume that cars can go from each city within the road network to any other city except to the city where they came from. This means that in a network with \( N \) cities we will have \( N \cdot (N-1) \) classes in our stochastic network and we denote \( K = N \cdot (N-1) \). For compactness we will number the classes from 1 up to \( N^2 \), where classes \( 1, N+2, 2N+3, \ldots, (N-2)N+(N-1) \), \( N^2 \) are dummy classes. This way, the number of the class can easily be interpreted. This is visualised in Table 6.1.

As we can see in Table 6.1, one can relate the number of a class to the origin-destination pair. One has to look at how many cities there are in total, in our example 5. In this example the number 17 can be rewritten as: \( 3 \cdot 5 + 2 \). The number in front of the number of cities +1, is the originating city for this class. The number that is left after division by the number of cities is the city number this class goes to. This means that class number 17 is the class that originated in city 4 and goes to city 2. Thus route \((i,j)\) corresponds to class \((i-1)N+j, i \neq j\). From a class \(k\) one can calculate \(i\) and \(j\) as in Equation (6.4).

\[
i = \left\lceil \frac{k}{N} \right\rceil
\]
\[
j = \begin{cases} N & \text{if } k \mod N = 0 \\ k \mod N & \text{otherwise} \end{cases}
\]

In our example from Figure 6.2 we have 25 classes with 5 dummy classes, see Table 6.2. We need this class numbering for the rest of the example.

**Table 6.1:** Numbering of customer classes, the red numbers are dummy classes.

<table>
<thead>
<tr>
<th>To/From</th>
<th>city 1</th>
<th>city 2</th>
<th>\ldots</th>
<th>city ( N-1 )</th>
<th>city ( N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>city 1</td>
<td>1</td>
<td>2</td>
<td>\ldots</td>
<td>( N-1 )</td>
<td>( N )</td>
</tr>
<tr>
<td>city 2</td>
<td>( N+1 )</td>
<td>( N+2 )</td>
<td>\ldots</td>
<td>( N+(N-1) )</td>
<td>2( N )</td>
</tr>
<tr>
<td>city 3</td>
<td>2( N+1 )</td>
<td>2( N+2 )</td>
<td>\ldots</td>
<td>2( N+(N-1) )</td>
<td>3( N )</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>city ( N-1 )</td>
<td>( (N-2)N+1 )</td>
<td>( (N-2)N+2 )</td>
<td>\ldots</td>
<td>( (N-2)N+(N-1) )</td>
<td>( (N-1)N )</td>
</tr>
<tr>
<td>city ( N )</td>
<td>( (N-1)N+1 )</td>
<td>( (N-1)N+2 )</td>
<td>\ldots</td>
<td>( (N-1)N+(N-1) )</td>
<td>( N^2 )</td>
</tr>
</tbody>
</table>

**Table 6.2:** Numbering of customer classes in our example, the red numbers are dummy classes.

**Arrivals** The arrivals in our model are the cars that leave a certain city to go to another city. This means that there are arrivals at each first queue of the road, since people from all cities and villages may
be going to another city. Note that by definition not all customer classes can arrive at each queue. A road is connected to two or more cities and therefore only classes originating in these cities are able to arrive on a certain road. The arrivals of class-\( k \) customers occur as independent Poisson processes with parameters \( \lambda_k \).

**Routing probabilities**  The routing probabilities in our model are class-dependent. This means that the probability of going in a certain direction is dependent on the destination of the car. The routing probabilities are determined beforehand and can be seen as an optimised way for cars to move throughout the network. For instance if a car can choose between two routes that on average take equally long, the routing probabilities can be divided 50/50.

We keep the routing probabilities in this section as general as possible, but only determine some of the probabilities that are by definition 0 or 1. As mentioned before, the other routing probabilities can be found in Chapter 8, where they are defined for a specific network.

All queues \( i \), where \( i \) is even, are second queues of a road. This means that there cannot be any arrivals to these queues. Besides, we know that a car that is in a first queue of a certain road will be routed towards the second queue of that same road with probability 1. The last probabilities that are certainly 0 are the arrival probabilities of certain classes to certain roads. As said before a class-\( k \) customer, where \( k = (i - 1)N + j \), originates from city \( i \). Therefore, the road where this class arrives at has to be linked to city \( i \).

The routing probabilities below will have a probability of 0 or 1.

- \( p_{0,i}(k) = 0 \), for all \( k \) and for all \( i \) even. This is because one cannot arrive at the second queue of a road.
- \( p_{i,i+1}(k) = 1 \), for all \( k \) and all \( i \) odd. From a first queue of the road you will always go to the second queue of the road.

Given a specific network, one can thus know some of the routing probabilities. We take a look at our example to show this. In Figure 6.3 we numbered all the servers on the roads from 1 to 12 and the cities from 1 to 5.
At first, we see that you will always go from system 1 to system 2, which means that $p_{1,2}(k) = p_{3,4}(k) = p_{5,6}(k) = p_{7,8}(k) = p_{9,10}(k) = p_{11,12}(k) = 1$, for all classes $k$.

Next, a class cannot arrive in a second system, which means: $p_{0,2}(k) = p_{0,4}(k) = p_{0,6}(k) = p_{0,8}(k) = p_{0,10}(k) = p_{0,12}(k) = 0$, for all classes $k$.

Other probabilities are known, because we see which cities are connected. For example from city 1 you can only go to server 1 and 3. This means that for arrivals we get that all classes that originate in city 1 cannot go to servers 5, 7, 9 and 11. The classes that originate in city 1 are classes $1, 2, 3, 4$ and 5. This means that for $k \in \{1, 2, 3, 4, 5\} : p_{0,i}(k) = 0$ for $i \in \{5, 7, 9, 11\}$. This same reasoning can also be applied to cities 2, 3, 4 and 5.

We also introduced some dummy classes that in the actual road network do not exist. These dummy classes never enter the entire queuing system, thus all arrival probabilities are zero.

All the known probabilities are listed below:

- From system 1 to system 2: $p_{1,2}(k) = p_{3,4}(k) = p_{5,6}(k) = p_{7,8}(k) = p_{9,10}(k) = p_{11,12}(k) = 1$, for all classes $k$.
- Not arriving in system 2: $p_{0,2}(k) = p_{0,4}(k) = p_{0,6}(k) = p_{0,8}(k) = p_{0,10}(k) = p_{0,12}(k) = 0$, for all classes $k$.
- Arriving probabilities from city 1: $p_{0,i}(k) = 0$, for $k \in \{1, 2, 3, 4, 5\}$ and $i \in \{5, 7, 9, 11\}$.
- Arriving probabilities from city 2: $p_{0,i}(k) = 0$, for $k \in \{6, 7, 8, 9, 10\}$ and $i \in \{3, 11\}$.
- Arriving probabilities from city 3: $p_{0,i}(k) = 0$, for $k \in \{11, 12, 13, 14, 15\}$ and $i \in \{1, 7, 9\}$.
- Arriving probabilities from city 4: $p_{0,i}(k) = 0$, for $k \in \{16, 17, 18, 19, 20\}$ and $i \in \{1, 3, 5, 9, 11\}$.
- Arriving probabilities from city 5: $p_{0,i}(k) = 0$, for $k \in \{21, 22, 23, 24, 25\}$ and $i \in \{1, 3, 5, 7\}$.
- Dummy classes: $p_{0,i}(k) = 0$, for all $i$ and $k \in \{1, 7, 13, 19, 25\}$.
Service  Since we have two separate queuing systems for each road we also have two separate service time distributions, service rates and service disciplines.

- Systems 1:
  - All first queues have a deterministic service time distribution with mean \( \beta_i, \) \( i \) odd. This mean is class-independent as all cars are identical.
  - \( r_i(n_i) = n_i \) for all \( i \) odd, since this queue has infinitely many servers.
  - The service discipline is a processor sharing service discipline, thus \( \phi_m(n) = \frac{1}{n} \) for all \( m = 1, \ldots, n. \)

- Systems 2:
  - All second queues have an exponential service time distribution with parameter \( \beta_i, \) \( i \) even.
  - These second queues are modelled as a single server queue, thus \( r_i(n_i) = 1 \) for all \( n_i \) and \( i \) even.
  - The service discipline is again a processor sharing service discipline, thus \( \phi_m(n) = \frac{1}{n} \) for all \( m = 1, \ldots, n. \)

Traffic equations  Since we have a lot of structure in our system, we can also see that the traffic equations can be simplified. For instance, we know that the number of queues, \( M, \) is even. We have two queues for all roads in our traffic system.

For all the second queues of a road, we know that all traffic comes from the first queue of that same road. This can also be seen in the traffic equations for these queues, see Equation (6.5).

\[
\gamma_i(k) = \gamma_{i-1}(k) \quad \forall i \text{ even} \tag{6.5}
\]

For the first queues of a road, the traffic equations have a more general form. We do know that not all classes can arrive at all queues, therefore \( p_{0,i}(k) = 0 \) for some pairs of \( k \) and \( i. \) However, this cannot be captured in a systematic way. On the other hand, we do know that the only traffic at a first queue comes from a second queue or from outside the network. Therefore, we can check only queues \( j, \) where \( j \) is even. We will give some traffic equations for our example in Equation (6.7), but for all queues \( i, \) where \( i \) is odd the traffic equation is given by Equation (6.6).

\[
\gamma_i(k) = \lambda_k p_{0,i}(k) + \sum_{2 \leq j \leq M_j \text{ even}} \gamma_j(k)p_{j,i}(k) \quad \forall k \tag{6.6}
\]

To show how these can be more simplified when the system is known, we will look at the traffic equations for queue 1 and 2 in our example (Figure 6.3). These traffic equations are written down in Equation (6.7). We make use of the fact that not all classes can arrive at queue 1 and the fact that customers in queue 12 first need to go to another road to end up in queue 1, thus \( p_{12,1} = 0. \)

\[
\begin{align*}
\gamma_1(k) &= \lambda_k p_{0,1}(k) + \sum_{j \in \{4, 6, 8, 10\}} \gamma_j(k)p_{j,1}(k) \quad \text{for } k \in \{2, 3, 4, 5, 6, 8, 9, 10\} \\
\gamma_1(k) &= \sum_{j \in \{4, 6, 8, 10\}} \gamma_j(k)p_{j,1}(k) \quad \text{for } k \in \{11, 12, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24\} \tag{6.7} \\
\gamma_1(k) &= 0 \quad \text{for } k \in \{1, 7, 13, 19, 25\} \\
\gamma_2(k) &= \gamma_1(k) \quad \forall k
\end{align*}
\]
Stability For stability in our system we have to check if \( \rho_i < r_i^* \) for all queues. Here \( \rho_i := \sum_{k=1}^{K} \rho_i(k) \), \( \rho_i(k) := \gamma_i(k)\beta_i(k) \) and \( r_i^* := \liminf_{n_i \to \infty} r_i(n_i) \).

For all first queues in our road we have that \( r_i(n_i) = n_i \), which means that \( r_i^* = \liminf_{n_i \to \infty} n_i = \infty \). Thus whatever \( \rho_i \) will be, the first queues will always be stable. The second queues are different, there \( r_i(n_i) = 1 \), thus \( r_i^* = \liminf_{n_i \to \infty} 1 = 1 \). Thus we know that \( \rho_i = \sum_{k=1}^{K} \rho_i(k) = \sum_{k=1}^{K} \gamma_i(k)\beta_i(k) < 1 \). We assume that the routing probabilities and service time distributions are chosen in such a way that this condition is met.

Stationary probabilities Now we look at the stationary probabilities. Recall that the state of the system can be described by a vector \( N(t) = (N_{ik}(t))_{i=1,...,M,k=1,...,K} \), where \( N_{ik}(t) \) denotes the total number of class-\( k \) customers present at queue \( i \) at time \( t \). The joint queue length has a product-form as shown in Equation (6.8).

\[
\pi(n) = \prod_{i=1}^{M} \phi_i(n_i)
\]

where

\[
\phi_i(n_i) = G_i^{-1} \frac{1}{\prod_{m=1}^{n_i} r_i(m)} \left( \frac{n_i + \cdots + n_K}{n_1 \cdots n_K} \right)^{n_i} \rho_i(1)^{n_1} \cdots \rho_i(K)^{n_K} 
\]

\[
G_i = \sum_{m_i=0}^{\infty} \frac{\rho_i^{m_i}}{m_i!} \frac{1}{\prod_{m=1}^{m_i} r_i(m)} 
\]

We see that we can already decompose the stationary probability per queue. Therefore, we will look at the first and second queues in the road individually.

For the first queues, i.e. \( i \) is odd, we know that \( r_i(n_i) = n_i \). Thus we can calculate \( G_i \), see Equation (6.9).

\[
G_i = \sum_{m_i=0}^{\infty} \frac{\rho_i^{m_i}}{m_i!} \frac{1}{\prod_{m=1}^{m_i} r_i(m)} 
\]

\[
\phi_i(n_i) = G_i^{-1} \frac{1}{\prod_{m=1}^{n_i} r_i(m)} \left( \frac{n_i + \cdots + n_K}{n_1 \cdots n_K} \right)^{n_i} \rho_i(1)^{n_1} \cdots \rho_i(K)^{n_K} 
\]

This means that \( \phi_i(n_i) \) can also be simplified, see Equation (6.10).

\[
\phi_i(n_i) = e^{-\rho_i} \frac{1}{\prod_{m=1}^{n_i} r_i(m)} \left( \frac{n_i + \cdots + n_K}{n_1 \cdots n_K} \right)^{n_i} \rho_i(1)^{n_1} \cdots \rho_i(K)^{n_K} 
\]

\[
= e^{-\rho_i} \frac{1}{\prod_{m=1}^{n_i} r_i(m)} \left( \frac{n_i + \cdots + n_K}{n_1 \cdots n_K} \right)^{n_i} \rho_i(1)^{n_1} \cdots \rho_i(K)^{n_K} 
\]

\[
= e^{-\rho_i} \frac{1}{n_i! \cdots n_K!} \rho_i(1)^{n_1} \cdots \rho_i(K)^{n_K} 
\]
For the second queues, i.e. \( i \) is even, we can do something similar, since we know \( r_i(n_i) = 1 \). \( G_i \) is calculated in Equation (6.11). Note that we know that \( \rho_i < 1 \) for these queues.

\[
G_i = \sum_{m_i=0}^{\infty} \frac{\rho_i^{m_i}}{\prod_{m=1}^{m_i} 1} = \sum_{m_i=0}^{\infty} \rho_i^{m_i}
\]

Including this in the equation for \( \phi_i(n_i) \) we get Equation (6.12).

\[
\phi_i(n_i) = G_i^{-1} \prod_{m=1}^{n_i + \cdots + n_i K} (n_{i1} + \cdots + n_i K) \rho_i(1)^{n_{i1}} \cdots \rho_i(k)^{n_i K} = (1 - \rho_i) \left( \frac{n_{i1} + \cdots + n_i K}{n_{i1} \cdots n_{iK}} \right) \rho_i(1)^{n_{i1}} \cdots \rho_i(k)^{n_i K}
\]

Together we can denote the joint queue length by Equation (6.13).

\[
\pi(n) = \prod_{1 \leq i \leq M, \ i \ odd} \left( e^{-\rho_i} \frac{\rho_i(1)^{n_{i1}} \cdots \rho_i(k)^{n_{iK}}}{n_{i1}! \cdots n_{iK}!} \right) \prod_{2 \leq i \leq M, \ i \ even} \left( 1 - \rho_i \right) \left( \frac{n_{i1} + \cdots + n_i K}{n_{i1} \cdots n_{iK}} \right) \rho_i(1)^{n_{i1}} \cdots \rho_i(k)^{n_i K}
\]

Equation (6.13) denotes the entire joint queue length distribution. We can use this to find the mean queue length, denoted by \( \mathbb{E}[L] \), in this system. With the mean queue length, we can calculate the mean sojourn time (\( \mathbb{E}[S] \)) by Little’s Law, see Equation (6.14).

\[
\mathbb{E}[L] = \lambda \mathbb{E}[S],
\]

where

\[
\lambda = \text{the total arrival rate into this system}
\]

### 6.3 Modelling of an intermodal network

An intermodal network in terms of a multi-class open stochastic network is a lot like a traffic network. Some of the links are actually roads, thus one only has to think about the other two modes, trains and barges. Therefore, we adjust some links in the example network from Figure 6.1, see Figure 6.4. The road from city 1 to city 3 is now a railway and the road from city 2 to city 5 is now waterway. The network in Figure 6.4 again serves as an example throughout this section.
Our multi-class open product-form network resembles the one previously described. The only thing we alter in this model is the queues for the trains and the barges. The arrival process and customer classes are all the same as in the previous model. As before, we want to model the travel times in the intermodal network as queues. Thus on the new links, we need to define new queues that model the behaviour of trains and barges. Trains and barges normally take a batch of containers at a certain time. Containers arriving too late have to wait for the next train or barge to be picked up. This kind of behaviour is called batch service.

In a singular queue, waiting time distributions in a queue with batch services can be calculated [84]. However, we want to add this queue to our already existing stochastic network. This makes the analytic analysis much more complicated. As discussed before, the product-from solution only holds in specific cases. For research in product-form networks with batch services, we refer to [61], [32] and [33]. However, most of these papers include the property that batches of customers collapse into one customer after being served at a queue. In an intermodal network this modelling choice is not desirable since our customers, the containers, might be routed differently after the queue with batch service. Another paper by Chao and Zheng [34], discusses a product-form solution for stochastic networks with batch arrivals and batch services without this property. However, these authors proof that the product-form solution only holds if an additional arrival process is turned on whenever a queue is empty. One can see that research on these networks is not necessary from a practical viewpoint, but mainly driven by the existence of the product-form. This means that we do not model the trains and barges as batch services, but try to get as close as possible.

Looking at an individual container that wants to travel with a train or barge, we can identify two periods. The first period is waiting until the train/barge arrives. The second period is the actual travelling on this train/barge. We discuss how we model these two periods separately in two queues. This means that the newly made links for trains and barges will also have two queues, see Figure 6.5. The two queues are described below. These queues are of course different from the queues on the normal roads, therefore we will number them as the third and fourth queuing systems in our network.

Figure 6.4: Example intermodal network.
• Third systems: the third systems model the waiting on a train/barge. Here we can think of a queue with infinite servers, since all containers immediately start with their waiting period. The service time is a uniform service time. The period of this uniform distribution is between 0 and the total time it takes a train/barge to make a full round. Therefore, we model that the container arrives uniformly in the time between a train/barge has just departed and the next train/barge arrives. This queue is again an infinite server queue, thus we work with a processor sharing discipline. This means $\phi_m(n) = \frac{1}{n}$ for all $m = 1, \ldots, n$ and $r(n) = n$. The service time distribution is an uniform distribution.

• Fourth systems: the fourth systems are the actually travelling on a train/barge. Obviously, a train or barge is capacitated, which means that if there are a lot of containers already waiting to go on a train and barge, some containers have to wait. The queue that models the travelling on a train/barge is therefore capacitated and has $c$ servers. This means that the service rate $r(n) = \max\{n, c\}$. The amount of servers of a queue is the capacity of the specific train/barge it is modelling. As discussed in Section 6.1, we need either a symmetric service discipline and a general service distribution or FCFS with an exponential service time. In this queue we would like to hold on to the FCFS property and therefore the service time distribution is an exponential service distribution. If we would not have a FCFS service discipline at this queue, it might occur that containers that arrive later are still able to catch a train/barge, while other containers (that arrived earlier) have to wait until the next train/barge. We assume all containers to be of equal priority and therefore, FCFS is the most fair policy.

These modelling choices show that the third systems are infinite server queues with an symmetric service discipline and the fourth systems are queues with a FCFS service discipline and an exponentially distributed service time. Therefore, the product-form solution for the stationary distribution described in Section 6.1 does still hold.
A lot of the parameters we described in the previous section, Section 6.2, still hold for an intermodal network. The classes, arrival processes, routing probabilities and traffic equations keep the same properties. We only need to check how the stability condition and the stationary probabilities differ in this intermodal network that includes trains and barges. For the first and second queuing systems in our network we do not redo the calculations, since these are the same as in Section 6.2.

**Stability** For stability in our system we again have to check if $\rho_i r_i^*$ for all queues. Here $\rho_i := \sum_{k=1}^K \rho_i(k), \rho_i(k) := \gamma_i(k)\beta_i(k)$ and $r_i^* := \lim_{n_i \to \infty} r_i(n_i)$.

For all third queues in our road we have again that $r_i(n_i) = n_i$. Therefore, these queues are stable. The fourth queues have $r_i(n_i) = \max\{n_i, c_i\}$, thus $r_i^* = \lim_{n_i \to \infty} \max\{n_i, c_i\} = c_i$. Thus we know that $\rho_i = \sum_{k=1}^K \rho_i(k) = \sum_{k=1}^K \gamma_i(k)\beta_i(k) < c_i$. We assume that the routing probabilities and service time distributions are chosen in such a way that this condition is met. Therefore, the system is stable.

**Stationary probabilities** For the stationary probabilities, we will again look at the individual queues, in particular the third and fourth queue of the system. The third queues are similar to the first queues, since $r_i(n_i) = n_i$. This means that again $G_i = e^{\rho_i}$ and $\phi(n_i)$ is given by Equation (6.15).

$$\phi_i(n_i) = e^{-\rho_i} \frac{\rho_i(1)^{n_{i1}}}{n_{i1}!} \cdots \frac{\rho_i(K)^{n_{iK}}}{n_{iK}!}$$  \hspace{1cm} (6.15)

The fourth queues are different, here $r_i(n_i) = \max\{n_i, c_i\}$. For these queues, calculating $G_i$ is a bit more difficult, since it has no nice form as the queues before. Therefore we will keep the stationary distribution for this queue in the original form, see Equation (6.16).

$$\phi_i(n_i) = G_i^{-1} \frac{1}{\Pi_{m=1}^{n_i+\cdots+n_{iK}} r_i(m)} \left(\frac{n_{i1} + \cdots + n_{iK}}{n_{i1} \cdots n_{iK}}\right)^{\rho_i(1)^{n_{i1}} \cdots \rho_i(K)^{n_{iK}}}$$.  \hspace{1cm} (6.16)

Together we can denote the joint queue length of the entire system by Equation (6.17). Again we can calculate the mean number of agents in the system and thus the mean sojourn time.

$$\pi(n) = \prod_{i \text{ is system } 1} \left(\frac{e^{-\rho_i} \rho_i(1)^{n_{i1}}}{n_{i1}!} \cdots \frac{\rho_i(K)^{n_{iK}}}{n_{iK}!}\right),$$

$$\prod_{i \text{ is system } 2} \left(\frac{1 - \rho_i}{n_{i1} \cdots n_{iK}}\right)^{\rho_i(1)^{n_{i1}} \cdots \rho_i(K)^{n_{iK}}} \frac{n_{i1} + \cdots + n_{iK}}{n_{i1} \cdots n_{iK}},$$

$$\prod_{i \text{ is system } 3} \left(\frac{e^{-\rho_i} \rho_i(1)^{n_{i1}}}{n_{i1}!} \cdots \frac{\rho_i(K)^{n_{iK}}}{n_{iK}!}\right),$$

$$\prod_{i \text{ is system } 4} \left(\phi_i(n_i) = G_i^{-1} \frac{1}{\Pi_{m=1}^{n_{i1}+\cdots+n_{iK}} r_i(m)} \left(\frac{n_{i1} + \cdots + n_{iK}}{n_{i1} \cdots n_{iK}}\right)^{\rho_i(1)^{n_{i1}} \cdots \rho_i(K)^{n_{iK}}}\right),$$
Chapter 7

Simulation-based models

As we have seen in the previous chapter, the analytic model involves several assumptions. The stochastic network is static, thus dynamic factors, such as dynamic routing, are not accounted for. Therefore, we also developed simulation-based models that can deal with more realistic scenarios. All these models aim to find the routes that minimise the costs for individual agents. This chapter describes the simulation used in these models and the models themselves. In this chapter, there is no difference in the analysis of a road network or a synchromodal network. Obviously, the costs for a road or for a train/barge are modelled differently, but this does not alter the analysis. We refer to a vehicle or container in the network as an agent. When analysing a traffic network, an agent is a vehicle and when analysing a synchromodal network, an agent is a container.

Firstly, in Section 7.1, we state the assumptions for all simulation-based models. Then, Section 7.2 describes the simulation we developed for the agents moving through a network. This simulation is then used in two simulation-based heuristics, which are approximations that try to find minimum-cost routes. The simulation is also part of a simulation-based solution algorithm, which finds an optimal user equilibrium for all agents. Section 7.4 describes the simulation-based heuristics. These heuristics assume that the agents in the network know the occupancy of the links in the network up to the current time. The heuristics aim to route the agents with the least amount of costs. Section 7.5 introduces a simulation-based solution algorithm that optimally assigns all agents to routes. This algorithm relies on the assumption that there is perfect knowledge. This means that all agents also know how many agents are due to arrive in the future. Obviously, in real life only estimates of these arrivals are known and therefore we extend the solution-based algorithm in Section 7.5.2 into a rolling-horizon approach. Lastly Section 7.6 discusses the correctness of these models and how they might be used in real synchromodal networks.

7.1 Assumptions

In the simulation-based models we assume certain properties as stated below.

Assumption 1. All nodes can be reached by truck.

Assumption 2. Waiting at intermediate nodes is not allowed.

Assumption 3. The costs of travelling links are non-negative.
**Assumption 4.** *All information from the network is available to all agents.*

For both simulation-based heuristics assumption 4 means that the occupancy on all links is known up to the departure time of the agent. For the simulation-based solution method, we assume full information. Therefore assumption 4 means that the occupancy on all links is known as well as future orders, i.e. containers that want to travel from an origin to a destination within the network. The routes that all agents take are also shared to the other agents in the simulation-based solution method. For the simulation-based solution algorithm within the rolling-horizon framework, we assume that the occupancy on the links is known up to the current time and the future estimates are known to all agents in the network.

In Section 7.3 we mention how we calculate time- and state-dependent travel times for our network. There are also some specific assumptions on the travel times.

**Assumption 5.** *The travel times of roads only depend on the occupancy of the link.*

**Assumption 6.** *The departure times of the trains and barges are known.*

**Assumption 7.** *The capacities of all trains/barges are known.*

**Assumption 8.** *The travel times of trains and barges over a certain link are constant.*

### 7.2 Description of simulation

The simulation is an event-driven simulation, where each new event triggers a change in the network. Possible events are listed below.

- **Request route:** a new agent asks for a route from an origin to a destination.
- **Enter link:** an agent will traverse a certain link in the network.
- **Leaving link:** when an agent reaches its destination or an intermediate node, he leaves the link.

The simulation handles the events one by one until a certain end time. The entire duration of the simulation is referred to as the planning horizon. One can see the pseudocode of the simulation in Algorithm 1. Here the event queue is a sorted queue, which holds all future events sorted by time. The handling of the events is discussed in detail below. In this simulation one can keep track of all kinds of performance measures: individual travel times, average travel times, occupancy on roads, etc. We discuss in Chapter 8 on which performance measures we focus on.
Algorithm 1 Simulation of events

1: \( t = 0 \)
2: while \( t < \) planning horizon do
3: \( \text{event} = \) first event from event queue
4: \( t = \) time of event
5: if \( \text{event} \) is an request route event then
6: Create path for this OD pair
7: Add enter link event for time \( t \) on first link
8: else if \( \text{event} \) is a enter link event then
9: Calculate travel time, \( t_T \)
10: Add leave link event for time \( t + t_T \)
11: Increase occupancy on this link by 1
12: if agent is not yet at its destination then
13: Add enter link event for the next link for time \( t + t_T \)
14: end if
15: else if \( \text{event} \) is an leave link event then
16: Decrease occupancy on this link by 1
17: end if
18: end while

Request route A container can ask for its route on multiple occasions. This could be an incoming order or the agent wants to take a different route to save costs. This means that the agent wants to know on which link he should be going. There might be multiple ways to calculate this route for the container and it may even change over time. However, we assume that there is a separate function handling this that returns the route of the incoming order. The heuristics and the simulation-based algorithm all have a function that can return a route given an origin, destination and current time.

Thus, the request of a route will be handled by asking for a route and creating an enter link event for the first link. This enter link event is at the same time as the route request. Note that this can easily be extended to a known departure time if orders come in before they have to be shipped.

Entering a link For an enter link event we know on which link the agent is travelling and we know the route of this agent. We need to calculate how long it will take the agent to traverse this link. The actual calculation of the state-dependent travel times will be discussed in Section 7.3, but again we assume there is a function returning this travel time.

An enter link event will be handled as follows; firstly, the travel time, denoted by \( t_T \), is calculated. We make a leaving link event for the current time plus this travel time. Furthermore, we keep track of the fact that an additional agent is on this link for the entire travel time duration. This influences the travel time for current and future agents in the network. Lastly, if this link was not the last link on the route of this agent, we create a drive event for the next link at time: current time +\( t_T \).

Leaving a link When an agent leaves a link, the only thing we have to alter is the occupancy on this link. In other words, we need to decrease the number of agents on this link by one, since this again influences the travel time for current and future agents that want to traverse this link.
7.3 State-dependent travel times

Since both the heuristics and the simulation-based solution algorithm need the calculation of the travel times, we describe our methods for these calculations. As mentioned before, the travel times are time- and state-dependent. However, in the heuristics and the solution algorithm we assume that we know how many agents occupy certain links and at what time they want to traverse the link. This means that the time and state is fixed for the calculation of the travel times. First, we elaborate on how we calculate the travel time for roads and next we elaborate how to calculate travel times for the rail and waterways.

**State-dependent travel times for roads.** In Dynamic Traffic Assignment (DTA) research the formula proposed by the Bureau of Public Roads (BPR) function is frequently used, see Equation (7.1).

\[
t(v) = t^0 \left(1 + 0.15 \left(\frac{v}{c}\right)^4\right)
\]

where

\[
t^0 = \text{free-flow travel time},
\]
\[
v = \text{amount of vehicles},
\]
\[
c = \text{capacity}.
\]

However, this function was proposed for use on freeways, used by cars. Since we also need to model trucks carrying containers, we want a formula that allows for a bit more tweaking by using more parameters. Akçelik [3] describes Davidson’s function. Davidson’s function is a general-purpose travel-time formula for transport planning purposes, see Equation (7.2).

\[
t = t_0 \left(1 + J_D \cdot \frac{x}{1 - x}\right)
\]

where

\[
t = \text{average travel time per unit distance},
\]
\[
t_0 = \text{minimum (free-flow) travel time per unit distance},
\]
\[
J_D = a \text{ delay parameter},
\]
\[
x = \frac{v}{C}, \text{ degree of saturation},
\]
\[
v = \text{demand flow rate (vehicles/hour)},
\]
\[
C = \text{capacity (vehicles/hour)}.
\]

In this same research by Akçelik, [3], certain issues with Davidson’s function are described. The parameters used by Davidson, particularly the delay parameter, lacks a clear interpretation. Also, the function does not have finite values if there are more cars on the road than its capacity can handle. This is not realistic, since cars will not actually be stopped at the beginning of a road, there will only be a traffic jam, increasing the average travel time per unit distance.
Therefore, Akçelik proposes a different time-dependent travel time function, described in Equation (7.3).

\[ t = t_0 + 0.25T_f \left( z + \left( z^2 + \frac{8J_A \cdot x}{C \cdot T_f} \right)^{0.5} \right) \]

where
\[
\begin{align*}
  t & = \text{average travel time per unit distance}, \\
  t_0 & = \text{minimum (free-flow) travel time per unit distance}, \\
  T_f & = \text{flow/analysis period}, \\
  J_A & = \text{a delay parameter}, \\
  z & = x - 1, \\
  x & = \frac{v}{C}, \text{degree of saturation}, \\
  v & = \text{demand flow rate}, \\
  C & = \text{capacity}.
\end{align*}
\]

This function assumes a constant demand pattern and no initial queue at the start of the flow period. The travel time is defined as experienced by all vehicles arriving during the specified flow period.

The author also proposes some parameters for this travel time function representing various road classes, see Table 7.1. Here \( v_0 = 1/t_0 \) and \( T_f = 1 \) hour.

<table>
<thead>
<tr>
<th>Description</th>
<th>( v_0 ) (km/h)</th>
<th>( C ) (veh/h/lane)</th>
<th>( J_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freeway</td>
<td>120</td>
<td>2000</td>
<td>0.1</td>
</tr>
<tr>
<td>Uninterrupted arterial road</td>
<td>100</td>
<td>1800</td>
<td>0.2</td>
</tr>
<tr>
<td>Interrupted arterial road</td>
<td>80</td>
<td>1200</td>
<td>0.4</td>
</tr>
<tr>
<td>Interrupted secondary road</td>
<td>60</td>
<td>900</td>
<td>0.8</td>
</tr>
<tr>
<td>High friction secondary road</td>
<td>40</td>
<td>600</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 7.1: Parameters for travel time functions representing various road classes [3].

We adjust these parameters for trucks using freeways in the Netherlands. Since the maximum velocity for trucks on freeways is 80 km/h, we choose the following parameters: \( v_0 = 80 \) and \( J_A = 0.1 \). The capacity is chosen with respect to the other parameters in our simulation. We refer to Chapter 8 for the full list of parameters.

**State-dependent travel times for rail and barges.** For trains and barges the travel time is calculated differently. A train or a barge will namely leave at certain departure times, which are known beforehand. At these departure times there is a maximum number of containers it can take. This means that whenever a container arrives at a link representing a train or a barge, we have to calculate the time it has to wait for the next train/barge and see if it still fits on the corresponding train or barge.

We assume that the departure times for the trains and barges are known. This is not a restrictive assumption, since the routing of the containers is mostly done after the planning of the schedules of the trains and barges. We also assume that for each link we know the capacity of the train or barge travelling that link. In our implementation the capacity is fixed for a certain link, but it can be easily extended that for each departure of a train/barge, the capacity is altered. This gives the possibility to model each train and barge separately with their individual departure time and capacity. Besides the departure...
time and capacity, we also need to know the travel time of the train/barge. Since the travel times for trains and barges are much less variable, we assume that they are constant in our implementation. This means that for each (rail/waterway) link we also know the time it takes to traverse that link. Again, we mention that the algorithm can be easily extended to a variant where the travel times for trains and barges are stochastic.

Let us assume we have a container at time $t$ that wants to travel over a link representing rail or waterway. We first look up the departure time of the first train/barge that will depart on this link. We denote this departure time with $t_D$. Then we look at the travel time for this train or barge, denoted by $t_T$. The travel time for the container at time $t$ is given by Equation (7.4).

$$\text{Travel time for a container at time } t = t_D - t + t_T$$  \hspace{1cm} (7.4)

However, in Equation (7.4) the capacity of the train/barge, denoted by $C$, is not yet accounted for. Therefore, we also have to take into account the number of containers that want to travel over this link on the current train/barge. As mentioned before we calculate the travel times with a known routing assignment, thus we know the number of agents that want to travel this link, denoted by $x$. For the first $C$ containers the travel time is given by Equation (7.4). The next $C$ containers have to wait for the next departure, $t_{D+1}$ and so on.

This is not the only information needed to calculate the travel times. Namely, we also have to account for the changes in time. For example, it might be that the next departure is 5 time steps away. Then we have to add all containers from these 5 time steps that want to traverse the link and see if that is still lower than the capacity $C$. The problem here is that the calculation of the travel time influences how many agents want to traverse the link and the other way around. Therefore, we made a choice in our algorithm that simplifies the calculation. If the train/barge wants to take on more containers than its capacity we will return a cost of $\infty$. This means that for an occupancy of $x$ containers on a rail/waterway link the travel time function is given by Equation (7.5). In our heuristics and the simulation-based solution algorithm, the eventual routing decisions are always within the capacity constraints due to this choice.

$$\text{Travel time for } x \text{ containers at time } t = \begin{cases} t_D - t + t_T & \text{for } x \leq C \\ \infty & \text{for } x > C \end{cases}$$  \hspace{1cm} (7.5)

One consequence of this simplification is that a container is not able to wait for a longer time period in order to catch a next train/barge. However, we want a container to keep moving and not wait at intermediate nodes. At roads the containers are also not able to wait for a less busy period. Waiting at intermediate nodes can improve the solution, but this is out of scope.

### 7.4 Simulation-based heuristics

Our simulation-based heuristics are loosely based on how drivers in a road network can adjust their route in current traffic. We assume that the knowledge of the network is available to all agents, but only up to the current time. This means that an agent does not know how the other agents are routed, but he does know where every agent is currently. This is quite realistic, since modern navigation systems already have traffic information. They gather information about how busy certain roads are and calculate the shortest route accordingly.

We have two simulation-based heuristics, one without and one with rerouting. This means that in the first heuristic we developed, all agents will know the state of the network and act upon this information.
However, once this choice has been made, it will not be altered anymore. This heuristic is explained in Section 7.4.1. In the second heuristic, explained in Section 7.4.2, agents can later decide to switch routes as conditions in the network change. Both heuristics rely on a dynamic shortest-path algorithm, which is described in Section 7.4.3.

7.4.1 Heuristic 1: minimum-cost routing without rerouting

The first heuristic aims to find the minimum-cost route in a greedy way. For all containers that request a route, the shortest path is calculated with a dynamic shortest-path algorithm. This dynamic shortest-path algorithm is described in Section 7.4.3. All containers that request a route, are thus given a route based on the current conditions. The containers will follow this route and do not adjust on a later time. The first heuristic is described in Algorithm 2. Note that this is the entire algorithm, including the simulation.

Algorithm 2 Heuristic 1: minimum-cost routing without rerouting

1: \( t = 0 \)
2: while \( t < \) planning horizon do
3: \( \text{event} = \) first event from event queue
4: \( t = \) time of event
5: if \( \text{event} \) is an request route event then
6: \( \text{Calculate shortest path under current conditions} \)
7: \( \text{Give this route to the current agent} \)
8: Add enter link event for time \( t \) on first link
9: else if \( \text{event} \) is a enter link event then
10: \( \text{Calculate travel time, } t_T \)
11: Add leave link event for time \( t + t_T \)
12: Increase occupancy on this link by 1
13: if agent is not yet at its destination then
14: \( \text{Add enter link event for the next link for time } t + t_T \)
15: end if
16: else if \( \text{event} \) is an leave link event then
17: Decrease occupancy on this link by 1
18: end if
19: end while

7.4.2 Heuristic 2: minimum-cost routing with rerouting

An improvement on the heuristic described in Section 7.4.1 is the rerouting of agents. As each agent traverses the network, it encounters intermediate nodes. However, for the simulation it does not matter if the agent arriving at that node arrived from outside the network or from another node. Therefore, we can recalculate the shortest path for each agent. Since the conditions in the network have likely changed, so may have the shortest path. This heuristic is described in Algorithm 3. Again this algorithm includes the simulation.
Algorithm 3 Heuristic 2: minimum-cost routing with rerouting

1: \( t = 0 \)
2: while \( t < \) planning horizon do
3: \( \text{event} = \) first event from event queue
4: \( t = \) time of event
5: if \( \text{event} \) is an request route event then
6: \( \text{Calculate shortest path under current conditions} \)
7: \( \text{Give this route to the current agent} \)
8: \( \text{Add enter link event for time } t \text{ on first link} \)
9: else if \( \text{event} \) is a enter link event then
10: \( \text{Calculate travel time, } t_T \)
11: \( \text{Add leave link event for time } t + t_T \)
12: \( \text{Increase occupancy on this link by 1} \)
13: if agent is not yet at its destination then
14: \( \text{Create an request route event for the next node at time } t + t_T \)
15: end if
16: else if \( \text{event} \) is an leave link event then
17: \( \text{Decrease occupancy on this link by 1} \)
18: end if
19: end while

7.4.3 Dynamic shortest-path algorithm

In these heuristics we have to calculate a shortest path again and again. Each event in the simulation might change something in the costs and thus the shortest paths might change after each event too. For a static network, a well-known technique like Dijkstra’s algorithm might be used. But in our network, one would have to repeat Dijkstra’s algorithm for each node, each time a condition within the network changes. However, running this algorithm each time there is a change in the network takes a lot of time. Researchers have tried to come up with more efficient dynamic algorithms that use the fact that not the entire network changes. One of these algorithms is described in the paper by Ramalingam and Reps [109]. The authors develop an incremental algorithm for the grammar problem which is a generalisation of the shortest-path problem. As a result they obtain a new dynamic single-source shortest-path problem, which can be extended to a dynamic all-pairs shortest-path problem.

We elaborate on Dijkstra’s algorithm first. The dynamic single-source shortest-path is essentially an extension of Dijkstra’s algorithm. In other words, when only inserting a collection of edges in an empty graph, the single-source shortest-path algorithm from [109] reduces to Dijkstra’s algorithm. We explain the algorithm described by Ramalingam and Reps thereafter. Note that this algorithm focuses only on changes within the network.

Dijkstra’s algorithm

The original algorithm by Edsger W. Dijkstra [48] found the shortest path in a network between two nodes. However, Dijkstra’s algorithm that is used more often nowadays fixes one node as a source node and calculates the shortest path between this source and all other nodes in the network. The algorithm is explained in Algorithm 4, where \( d(u) \) is a cost function for vertex \( u \) and \( s \) is the source node.
Algorithm 4 Dijkstra’s algorithm

1: Create a set $S$ that keeps track of all vertices whose minimum distance is found; initially $S = \emptyset$
2: Assign $d(u) = \infty$ for all vertices $u$; except for $s$, $d(s) = 0$
3: while $S$ does not include all vertices of the graph do
4: Pick a vertex $u$, which is not in $S$ and has minimum distance value
5: Include $u$ in $S$
6: Update distance value of all vertices adjacent to $u$. This can be done by iterating through all adjacent vertices $v$. The update is described by the Bellman-Ford equation: $d(v) = \min \{d(u) + \text{weight of edge } u \rightarrow v, d(v)\}$
7: end while

Dynamic all-pairs shortest-path algorithm

The incremental algorithm described in the paper by Ramalingam and Reps [109], is a solution algorithm to the grammar problem. This grammar problem is a generalisation of the shortest-path problem. However, since we only need the algorithm to solve the shortest-path problem, we will limit ourselves in this section to explain the algorithm already applied to the shortest-path problem.

Background A directed graph $G = (V(G), E(G))$ consists of a set of vertices $V(G)$ and a set of edges $E(G)$. A directed edge from $u$ to $v$ is denoted by $u \rightarrow v$. $u$ is a predecessor of $v$ and $v$ is a successor of $u$. A set of all predecessors of a vertex $u$ is denoted by $\text{Pred}(u)$ and the set of all predecessors of a set of vertices $U$ is denoted by $\text{Pred}(U) = \bigcup_{u \in U} \text{Pred}(u)$. The set of all successors of a vertex $u$ is denoted by $\text{Succ}(u)$ and the set of all successors of a set of vertices $U$ is denoted by $\text{Succ}(U) = \bigcup_{u \in U} \text{Succ}(u)$. In shortest-path problems every edge $u \rightarrow v$ has an associated length denoted by $\text{length}(u \rightarrow v)$. The length of a path in the graph is defined to be the sum of the lengths of the edges from that path.

We first focus on the single-source shortest-path problem (SSSP). In this problem, we need to determine for a specified source vertex, denoted by source($G$), the shortest path to each vertex $u$, $u \in G$. We assume that each edge in $G$ has a non-negative length.

The SSSP problem induces a set of equations. These equations are referred to as the Bellman-Ford equations. The Bellman-Ford equations introduce a distance measure for every vertex $u \in G$. They define the costs of the shortest path from source($G$) to $u$, denoted by $d(u)$. The equations are described in Equation (7.6).

$$d(u) = \begin{cases} 0 & \text{if } u = \text{source}(G) \\ \min_{v \in \text{Pred}(u)} \{d(v) + \text{length}(v \rightarrow u)\} & \text{otherwise} \end{cases} \quad (7.6)$$

The maximal fixed point of this set of equations is the solution to the SSSP problem [59]. If the edge lengths are all positive, the above set of equations has a unique fixed point. This means that for graphs with positive edge lengths, we can view the SSSP problem as the problem to solve Equation (7.6).

Idea of the algorithm The algorithm described in the paper by Ramalingam and Reps [109] has a set of equations as input. Assume that the set of equations consists of $k$ unknowns: $x_1, \ldots, x_k$ and $x_i = g_i(x_1, \ldots, x_k)$. These equations have to be strict weakly superior functions (SWSF), which means they have to be monotone nondecreasing in each variable and for every $i$:

$$g(x_1, \ldots, x_i, \ldots, x_k) < x_i \rightarrow g(x_1, \ldots, x_i, \ldots, x_k) = g(x_1, \ldots, \infty, \ldots, x_k)$$
Important to note is that the Bellman-Ford equations are SWSF functions. In our example \( x_i \) is the \( i \)-th vertex and \( g_i(x_1, \ldots, x_k) = \min_{x \in \text{Pred}(x_i)} [d(x) + \text{length}(x \rightarrow x_i)] \) if \( x_i \) is not the source vertex.

We start with some notation on outputs:

- \( d[x_i] \), this is a tentative output, which is the value of vertex \( x_i \) throughout the algorithm.
- \( d^*(x_i) \), this is the actual output that vertex \( x_i \) should have in the fixed point.
- \( \text{rhs}(x_i) \), this value is defined as \( g_i(d[x_1], \ldots, d[x_k]) \). So it is the value of the right-hand side of the equation of vertex \( x_i \) under the given assignment of values.

We define the terms consistent, inconsistent, over-consistent and under-consistent.

**Definition 1.** A vertex \( x_i \) is consistent if \( d[x_i] = \text{rhs}(x_i) \)

**Definition 2.** If \( d[x_i] \neq \text{rhs}(x_i) \) for a vertex \( x_i \), this vertex is inconsistent.

There are two types of inconsistent vertices:

**Definition 3.** If \( d[x_i] > \text{rhs}(x_i) \), then \( x_i \) is over-consistent.

**Definition 4.** If \( d[x_i] < \text{rhs}(x_i) \), then \( x_i \) is under-consistent.

A vertex \( x_i \) is correct when \( d[x_i] = d^*(x_i) \). However, during the algorithm \( d^*(x_i) \) is not known, thus the algorithm has to work with the consistency statuses. Figure 7.1 shows an example of how the deletion and insertions of edges can create both under- and over-consistent vertices.

![Figure 7.1](image-url)  
Figure 7.1: Example of over-consistent and under-consistent vertices in the dynamic SSSP problem. The figure on the left indicates a graph for which the single-source shortest-path information has been computed. All vertices are consistent in this graph. The simultaneous deletion of the edge \( \text{source} \rightarrow b \) and the insertion of the edge \( e \rightarrow c \) make vertex \( b \) under-consistent and vertex \( c \) over-consistent. Observe that though \( c \) is inconsistent its value is correct. [109]

The idea of the algorithm is to process the inconsistent vertices in the “correct” order. Each vertex will only be handled at most twice. The authors show that the “correct” order to process the inconsistencies is in increasing order of \( \text{key} \), where the key is defined in Equation (7.7).

\[
\text{key}(x_i) = \min (d[x_i], \text{rhs}(x_i)) 
\] (7.7)
This means that the key of an over-consistent vertex $x_i$ is $\text{rhs}(x_i)$ and the key of an under-consistent vertex $x_i$ is $d[x_i]$. The authors prove that if $x_i$ is the inconsistent vertex with the least key, then $x_i$ is guaranteed to be an overestimated vertex if it is over-consistent, and it is guaranteed to be an underestimated vertex if it is under-consistent. They also show that if an inconsistent vertex with the least key is over-consistent, then the $\text{rhs}$ value of this vertex is the correct value. However, this is not true for under-consistent vertices. Thus if $x_i$ is an under-consistent vertex with the least key we will process this vertex by first setting the value to $\infty$, making it an over-consistent vertex, and then process it as an over-consistent vertex. The idea of the algorithm is illustrated in Algorithm 5, which is from [109]. (Note that $\text{rhs}$ and $\text{key}$ are functions.) The authors prove that the algorithm is correct, it does not change the value of any unaffected vertex, and it changes the value of a vertex at most twice.

**Algorithm 5** Incremental fixed-point algorithm after a change in the set of equations.

**Require:** $Q$ a set of SWSF equations

1. $\text{rhs}(u) = g_u(d[x_1],\ldots,d[x_k])$
2. $\text{key}(u) = \min(d[u],\text{rhs}(u))$
3. while there exist inconsistent variables in $Q$ do
4. Let $u$ be an inconsistent vertex with minimum $\text{key}$ value
5. if $\text{rhs}(u) < d[u]$ then \{$u$ is over-consistent\}
6. $d[u] = \text{rhs}(u)$
7. else if $d[u] < \text{rhs}(u)$ then \{$u$ is under-consistent\}
8. $d[u] = \infty$
9. end if
10. end while

**Algorithm** A detailed version of the pseudocode of the algorithm can be found in Algorithm 6. It is assumed that the set of vertices $U$ includes all modified vertices and so all vertices in $V(G) - U$ are consistent.
Algorithm 6 An algorithm for the dynamic fixed-point problem

Require: $G$: a dependence graph of a set of SWSF equations, $U$: the set of modified vertices in $G$, $u, v, w$: vertices, Heap: a heap of vertices

Ensure: Every vertex in $V(G)$ is consistent

1: Heap = $\emptyset$
2: for $u \in U$ do
3:   $\text{rhs}[u] = g_u(d[x_1], \ldots, d[x_k])$
4:   if $\text{rhs}[u] \neq d[u]$ then
5:     InsertIntoHeap(Heap, $u$, min($\text{rhs}[u], d[u]$))
6:   end if
7: end for
8: while Heap $\neq \emptyset$ do
9:   $u = \text{ExtractAndDeleteMin}(\text{Heap})$
10:   if $\text{rhs}[u] < d[u]$ then \{ $u$ is over-consistent \}
11:       $d[u] = \text{rhs}[u]$
12:   else \{ $u$ is under-consistent \}
13:       $d[u] = \infty$
14:   end if
15:   for $v \in (\text{Succ}(u) \cup \{u\})$ do
16:     $\text{rhs}[v] = g_v(d[x_1], \ldots, d[x_k])$
17:     if $\text{rhs}[v] \neq d[v]$ then
18:       AdjustHeap(Heap, $v$, min($\text{rhs}[v], d[v]$))
19:     else if $v \in \text{Heap}$ then
20:       Remove $v$ from Heap
21:   end if
22: end for
23: end while

For efficiency all inconsistent vertices are maintained in a min-heap. In a min-heap, the element with the lowest key is always stored in the root and therefore, it is very fast in finding and extracting this element. Also the value of $\text{rhs}[u]$ is maintained for every inconsistent vertex $u$. The heap operations are the following:

- $\text{InsertIntoHeap}(H, i, k)$ inserts $i$ into min-heap $H$ with key $k$.
- $\text{ExtractAndDeleteMin}(H)$ returns the vertex in min-heap $H$ with the minimum key and deletes it.
- $\text{AdjustHeap}(H, i, k)$ inserts $i$ into min-heap $H$ with key $k$ if $i$ is not in the min-heap, and changes the key of $i$ to $k$ if $i$ was already in the min-heap.

In our simulation we should keep a list of all vertices that undergo change, so at each time we need a shortest path, we can call this algorithm. However, since we are not focused on optimising the running time yet, we just call this algorithm with all vertices as an input. This induces only extra computation time in the first few lines of the algorithm, since consistent vertices will never be placed in the min-heap.

7.5 Simulation-based solution algorithm

The simulation-based solution algorithm calculates the optimal route for each agent in a certain planning horizon. We explain how the algorithm works in our agent-centric network.
In Section 7.5.1 we describe how for a given planning horizon with full information, a user equilibrium is reached between all agents. This means that no agent can alter his/her route in order to decrease his/her travel time. Here we assume that for the given planning horizon we know all information. Thus we also know how many agents will arrive in the future. In Section 7.5.2, we explain how we can alter this algorithm to a quasi real-time algorithm in which we do not assume that we have full information, but short-term and long-term estimates.

### 7.5.1 Solution for user equilibrium routing.

For all classes of agents in the network, i.e. all origin-destination pairs, we know all arrivals in the planning horizon. For each of the links we should know the time- and state-dependent travel time function. These travel time functions are already discussed in Section 7.3. We should also know the costs of traversing a link. We focus on the user equilibrium case which is defined in Definition 5 (Wardrop’s UE condition [132]).

**Definition 5.** The system has a user equilibrium if no agent can improve his/her experienced travel time by unilaterally switching routes (for a given departure time).

The simulation-based solution algorithm is an iterative procedure which switches between a simulation of events under given route assignments and calculating new route assignments for differences in the travel times and costs. In the first step we calculate the shortest paths for all origin-destination pairs when no traffic traverses the links. Then we will simulate the arrivals of the agents in the planning horizon and send everybody on these shortest paths. However, by sending agents over the links, the travel times and costs will be altered. The simulation will calculate the new travel times and log how many agents are on each link of the network for the entire planning horizon. This information is then used to recalculate the shortest paths. By rerouting the agents in each iteration we want to reach a fixed point in this system. In this fixed point the travel times and costs are not altered anymore and thus also the shortest paths are not altered anymore. Kaufman et al. [72] describe that this fixed point solution satisfies the UE condition.

**Simulation-based solution algorithm.** In Figure 7.2 one can see the global idea of our solution. The steps of the algorithm are described in further detail below. The algorithm is based on the algorithm for traffic networks described by Peeta and Mahmassani [98].
Simulate movement
Optimal route choice
assignment
Calculate state- and time-dependent link travel times
Shortest path algorithm
All-or-nothing assignment to shortest paths $S_i$
Update route assignment:
$$R_i = \frac{1}{i} S_i + \left(1 - \frac{1}{i}\right) R_{i-1}$$

Reached new user equilibrium?

Feasible paths: $R_0$

Optimal route choice assignment

Figure 7.2: Flow chart describing the general solution algorithm.

- **Step 1.** The iteration counter $i$ is set to 0. For all origin-destination pairs and all time steps we calculate feasible paths. We used Dijkstra’s algorithm to create these paths with the free-flow travel times and costs. The first route assignment is sending all incoming agents over their shortest path. This route assignment is denoted by $R_0$

- **Step 2.** Perform simulation of the planning horizon by sending the agents over their path assignments $R_i$. In this simulation we log the changes throughout time, obtaining the number of agents on each link on each time. We also keep track of the individual travel times of the agents.

- **Step 3.** Compute the new time- and state-dependent travel times with the information gathered from the simulation. How we calculate these travel times is described in Section 7.3.

- **Step 4.** Compute the new time-dependent shortest paths for all origin-destination pairs. The algorithm is described in Section 7.5.1.

- **Step 5.** We create an auxiliary route assignment by the *all-or-nothing assignment*. The all-or-nothing assignment basically sends all agents from a certain origin-destination pair that want to depart at a certain time step over the same route. The auxiliary route assignment we create, denoted by $S_i$, is sending all agents on their shortest path calculated in the previous step.

- **Step 6.** Then we calculate the new route assignment. The new route assignment is a combination between the old route assignment and the auxiliary route assignment calculated in the previous step. A new route assignment is calculated with the use of the Method of Successive Averages (MSA), see Equation (7.8). This equation is clarified in Section 7.5.1.

$$R_i = \frac{1}{i} S_i + \left(1 - \frac{1}{i}\right) R_{i-1}$$ (7.8)
• **Step 7.** The convergence criterion is based upon how much the occupancy on the links changes from one simulation to the next. We keep the log from the previous simulation and the one from this simulation and check the difference in occupancy on each link. If this difference is less than 5%, the convergence criterion is met.

• **Step 8.** If the convergence criterion is met, terminate the algorithm. Otherwise repeat from step 2 with the new route assignments.

It is important to know that the shortest-path algorithm in step 4 is discrete and therefore the planning horizon is divided in multiple time steps. At the end of the algorithm we know the shortest paths for all agents at each time step. To cope with this fact in step 2, we group all arrivals in the same time step and handle them as if they all occur at the beginning of this time step.

In the upcoming sections we elaborate on parts of our solution algorithm. Section 7.3 already handled the time- and state-dependent travel time functions we use in our solution algorithm. In Section 7.5.1 we discuss the shortest path algorithm we use and in Section 7.5.1 we discuss the method of successive averages in more detail.

**Time-dependent shortest-path algorithm** To find the time-dependent shortest path in step 4 from our solution algorithm we used the algorithm proposed by Ziliaskopoulos and Mahmassani [138]. This algorithm calculates the time-dependent shortest paths from all nodes in a network to a given destination node (denoted by \( N \)). Note that this is a discrete-time algorithm, thus it calculates the shortest paths for each time step over a given time horizon. It is based on Bellman’s principle of optimality.

**Terminology and notation** Denote by \( G = (V, E) \) a graph where \( V \) is the set of nodes and \( E \) is the set of edges. \( d_{ij}(t) \) is the non-negative time required to travel from node \( i \) to node \( j \), when the departure time is \( t \). We also have a cost function \( c_{ij}(t) \) that is the non-negative cost to travel from node \( i \) to node \( j \) for departure time \( t \). A time step is denoted by \( \delta \) and the amount of time steps with \( M \). The first possible travel time is \( t_0 \). This means that the set \( S \) denotes the set of all discrete time-stamps regarded in this algorithm, \( S = \{t_0, t_0 + \delta, t_0 + 2\delta, \ldots, t_0 + M\delta\} \).

\( \lambda_i(t) \) denotes the total costs of the current shortest path from node \( i \) to node \( N \) at time \( t \). \( \Lambda_i \) is the vector of all lambdas for node \( i \) at each time step, \( \Lambda_i = [\lambda_i(t_0), \lambda_i(t_0 + \delta), \ldots, \lambda_i(t_0 + M\delta)] \). This means that \( \Lambda_i \) is a vector of length \( M \). \( \lambda_i(t) \) is defined by Equation (7.9).

\[
\lambda_i(t) = \begin{cases} 
\min_{j \neq i} \{c_{ij}(t) + \lambda_j(t + d_{ij}(t))\} & \text{for } i = 1, 2, \ldots, N - 1; t \in S \\
0 & \text{for } i = N; t \in S
\end{cases}
\]

(7.9)

We remark that this definition is slightly different from the one in [138]. The reason for this is that we want to work with costs and not only travel times. For example, consider a path from \( i \) to \( j \). Starting at node \( i \) at time \( t \) link \( i \rightarrow k \) is chosen as the next link on the path based on the costs \( c_{ik}(t) \). However, node \( k \) is reached at time \( t + d_{ij}(t) \) and not \( t + c_{ij}(t) \). Therefore, the costs considered on the second link, \( k \rightarrow j \), is \( c_{kj}(t + d_{ij}(t)) \).
Assumptions on $c_{ij}(t)$ Since this algorithm works in discrete time and up to a certain planning horizon, it might occur that an agent arrives at a node in between time stamps or after the end time. This means that we have to make assumptions about the costs of travelling links in between the time stamps. Therefore we use the following assumptions:

1. $c_{ij}(t)$ for $t > t_0 + M\delta$ is constant and equal to $c_{ij}(t_0 + M\delta)$
2. $c_{ij}(\tau) = c_{ij}(t_0 + k\delta)$ for every $\tau$ in the interval \{ $t_0 + k\delta, t_0 + (k + 1)\delta$ \}

The first assumption says that after the considered time horizon, the costs of travelling the links will stay constant to the last value. One can always increase the time horizon if this assumption proves to be too restrictive. The second assumption deals with travelling in between time stamps, where we assume the costs of travelling are equal to the costs at the beginning of the time stamp.

Algorithm The definition of $\lambda_i(t)$ in Equation (7.9) is the main idea of the algorithm. However, instead of scanning all nodes in each iteration, a list is maintained which includes nodes with the potential to improve the costs. We refer to this list as the scan eligible (SE) list. Algorithm 7 describes the time-dependent shortest-path algorithm, this algorithm is from [138].

The steps of the algorithm are:

1. Create the SE list and insert the destination node $N$. Initialise the label vectors as follows: $\Lambda_N = (0, 0, \ldots, 0)$ and $\Lambda_i = (\infty, \infty, \ldots, \infty)$ for $i \in \{1, 2, \ldots, N - 1\}$.
2. Select and delete the first node from the SE list, defined to be node $i$. Scan the current node $i$ according to the Equation (7.10).

$$\lambda_j(t) = \min \{ \lambda_j(t), c_{ji}(t) + \lambda_i[t + d_{ji}(t)] \} \quad \text{for} \quad j \in \Gamma^{-1}(i)$$

where $\Gamma^{-1}(i)$ is the set of nodes that can directly reach $i$ (7.10)

Specifically, for every time step $t$, check whether $\lambda_j(t)$ is greater than $c_{ji}(t) + \lambda_i[t + d_{ji}(t)]$. If it is, replace $\lambda_j(t)$ in $\Lambda_j$ at position $i$ with the new value. If at least one of the labels of node $j$ has improved, insert node $j$ in the SE list.

If the SE list is empty, go to step 4.

3. Repeat step 2.

4. Terminate the algorithm.
Algorithm 7 Algorithm for time-dependent shortest path

1: Call \texttt{Creation}
2: while SE list is not empty do
3: \hspace{1em} Call \texttt{Deletion}(currentNode)
4: \hspace{2em} for All nodes \( j \) that can directly reach currentNode do
5: \hspace{3em} \texttt{nextNode} = \( j \)
6: \hspace{2em} \texttt{InsertInSEList} = false
7: \hspace{2em} for \( t = 1, \ldots, M \) do
8: \hspace{3em} \texttt{CurrentTravelTime} = \( d_{currentNode, nextNode}(t) \)
9: \hspace{3em} \texttt{NewLabel} = LABEL(currentNode, \( t \) + \texttt{CurrentTravelTime}) + \( c_{currentNode, nextNode}(t) \)
10: \hspace{3em} if LABEL(nextNode, \( t \)) > \texttt{NewLabel} then
11: \hspace{4em} LABEL(nextNode, \( t \)) = \texttt{NewLabel}
12: \hspace{4em} \texttt{InsertInSEList} = true
13: \hspace{4em} PathPointer(nextNode,\( t \)) = currentNode
14: \hspace{3em} end if
15: \hspace{2em} end for
16: \hspace{2em} if InsertInSEList then
17: \hspace{3em} Call \texttt{Insertion}(nextNode)
18: \hspace{2em} end if
19: \hspace{2em} end for
20: \hspace{1em} end while

Upon termination of the algorithm the \( M \)-dimensional vectors \( \Lambda_i \) (called LABEL in the pseudocode) contain the costs of the time-dependent shortest paths from every node \( i \) to the destination node \( N \) for each time step \( t \). The Path Pointer from Algorithm 7 refers to the next node in the shortest path.

For the implementation of the SE list the authors of [138] decided upon the data structure \texttt{deque} and implemented it as a one-dimensional array. This array holds an integer number and can have the values defined in Equation (7.11).

\[
\text{Deque}(i) = \begin{cases} 
-1 & \text{if node } i \text{ was in the list, but it is not now, } \\
0 & \text{if node } i \text{ has never been in the SE list, } \\
j & \text{if node } i \text{ is currently in the list and } j \text{ is the next node in the list, } \\
\infty & \text{if node } i \text{ is the last node in the list.}
\end{cases}
\tag{7.11}
\]

A deque has the following operations: creation, insertion and deletion. There will also be two pointers, one to the first element in the deque ("FIRST") and one to the last element in the deque ("LAST"). The pseudocode for these operations can be found in Algorithms 8, 9 and 10.

- **Creation.** This initialises the deque. We want to only insert the destination node \( N \) in the deque and all others will be initialised to 0, because they have never been in the deque yet. This means that this routine will set \( \text{Deque}(i) = 0 \) for \( i = 1, 2, \ldots, N - 1 \) and \( \text{Deque}(N) = \infty \). The pointers will also be set to \( N \), thus \( \text{FIRST} = \text{LAST} = N \).

- **Insertion.** A new node will be inserted at the beginning or at the end of the deque. This depends on whether or not the node has already been in the list or not. It will be inserted at the end of the deque when it is the first time for a node to be inserted. If it was in the list before, it will be inserted at the beginning. This routine does nothing if the node is already in the list. Thus for insertion of node \( i \) we get:
  - If \( \text{Deque}(i) = 0 \), set \( \text{Deque}(i) = \infty \) and \( \text{LAST} = i \).
If Deque(i) = −1, set Deque(i) = FIRST and FIRST = i.
- Else do nothing.

We remark that it is possible for a node to be inserted in an empty list, while the algorithm has not yet finished. This occurs when the first node of the list is deleted and then we check all neighbouring nodes. If one of them has the possibility to improve, we want to add it to the list. If the deleted node was the only node in the deque, the deque is empty in between these steps. Therefore, we altered the insertion routine slightly from the routine described in the paper. If the deque is empty at the beginning of the insertion, we set: Deque(i) = ∞ and the pointers FIRST = LAST = i.

• **Deletion.** This routine deletes the first element from the list and returns it. Thus the pointer “FIRST” is set to the second element in the list, which is equal to Deque(FIRST). It also changes the value of the deleted node to −1. For example, node 2 is the first element in the list. Then it will set FIRST = Deque(2) and Deque(2) = −1.

**Algorithm 8** Creation
1: for i = 1,...,N − 1 do
2: Deque(i) = 0
3: end for
4: Deque(N) = ∞
5: FIRST = N
6: LAST = N

**Algorithm 9** Insertion(i)
1: if Deque is empty then
2: Deque(i) = ∞
3: FIRST = i
4: LAST = i
5: else
6: if Deque(i) = 0 then
7: Deque(LAST) = i
8: LAST = i
9: Deque(i) = ∞
10: else if Deque(i) = −1 then
11: Deque(i) = FIRST
12: FIRST = i
13: end if
14: end if

**Algorithm 10** Deletion(currentNode)
1: currentNode = FIRST
2: FIRST = Deque(currentNode)
3: Deque(currentNode) = −1

**Method of successive averages** In our solution algorithm in step 6, we use the method of successive averages to make new route assignments for the agents in each iteration step.

First, we explain why sending all agents over their new shortest paths would not always result in convergence. For example, we have a network consisting of only two nodes and two links, see Figure 7.3.
Figure 7.3: Example: iterative method alone does not converge to user equilibrium.

Assume links 1 and 2 both have a cost function that increase with their load, e.g. $T_i = 6 + \frac{v}{10}$ for link $i$, $i = 1, 2$, where $v$ is the number of agents on this link. Let us say 90 vehicles want to traverse from node 1 to node 2 at a certain time step. At first all vehicles will travel via link 2 since without vehicles on the links, the costs for link 1 are 6 and the costs for link 2 are 3. This will result in new costs: 6 for link 1 and 12 for link 2. This means that in the next iteration, all agents will move to link 1, which creates the costs: 15 for link 1 and 3 for link 2. Thus all agents will switch again to link 2, etc. Obviously, the optimal policy is to split the agents: $\frac{1}{3}$ over link 1 and $\frac{2}{3}$ over link 2. This will result in the costs: 9 for link 1 and 9 for link 2, which satisfies the UE condition.

The method of successive averages is widely used in iterative algorithms. In each iteration of the algorithm, the current solution is averaged with a new solution generated by the algorithm, in our case the new shortest paths. Weights can be used to favour one of the two solutions. These weights are referred to as the step size. Typically, the step size is determined as a sequence that decreases to zero, e.g. $\frac{1}{i}$, where $i$ is the iteration index. This is a typical choice, since in each iteration the old solution becomes more and more reliable, since we are converging to a fixed point.

In our solution algorithm we want to use the route assignments we found with the shortest-path algorithm ($S_i$) and the route assignment of the previous iteration ($R_{i-1}$). Peeta and Mahmassani use the same iteration method in [98]. The new route assignment of the $i$-th iteration is given by Equation (7.12).

$$R_i = \frac{1}{i} S_i + \left(1 - \frac{1}{i}\right) R_{i-1}$$ (7.12)

If we take a look at Figure 7.3, we can describe what the method of successive averages will do in this example.

- **Iteration 1**: We first send all agents over the shortest paths, thus link 2. In other words $R_1$ is sending everyone over link 2.

- **Iteration 2**: The new costs are calculated: 6 for link 1 and 12 for link 2. This means that the new shortest path is link 1 and thus $S_2$ is sending all agents over link 1. This means for route assignment $R_2$:

$$R_2 = \frac{1}{2} S_2 + \left(1 - \frac{1}{2}\right) R_1$$

Thus $R_2$ sends half the agents over link 1 and half the agents over link 2.
• **Iteration 3:** Then we get new travel times: 10.5 for link 1 and 7.5 for link 2. The shortest path is link 2 and thus $S_3$ is sending all agents over link 2. Then for route assignment $R_3$:

$$R_3 = \frac{1}{3} S_3 + \left( 1 - \frac{1}{3} \right) R_2$$

Thus we send $\frac{1}{3}$ over link 2 and $\frac{2}{3}$ will be split in half over link 1 and link 2. In total we get $\frac{1}{3}$ over link 1 and $\frac{2}{3}$ over link 2.

• **Iteration 4:** The new costs are 9 for both links.

### 7.5.2 Rolling-horizon framework

In the algorithm described in Section 7.5.1, we assume that we know the information for the entire planning horizon. This means also knowing the future arrivals of agents in the network. Realistically, one would only have estimates for these arrivals. Peeta and Mahmassani expand their solution into a rolling-horizon solution framework [100]. The algorithm still tries to find an optimal UE routing, but uses short-term and medium-term forecasts for future arrivals. The rolling-horizon approach uses the algorithm discussed in [98] as a subroutine. We apply the same technique to the solution algorithm discussed in Section 7.5.

**The rolling-horizon approach** Figure 7.4 shows the rolling-horizon approach. The stage length of $h$ units in this picture refers to the planning horizon from the algorithm described in Section 7.5.1. This stage length is divided into small time intervals. For the first $l$ time intervals reliable short-term forecasts are known. Forecasts for the rest of the stage length are also known, but may be less reliable.

![Figure 7.4: The rolling-horizon approach for a quasi-real time assignment problem, from [100].](image-url)
The algorithm works in different stages, in Figure 7.4 two stages are illustrated. In stage $\sigma - 1$ the route assignments are calculated for the entire stage length of $h$ units, but only implemented for the first $l$ units. Then we will roll our horizon $l$ time steps forward and adjust the forecasts accordingly, this will bring us in stage $\sigma$. We have a new set of reliable short-term and less reliable medium-term forecasts for the next $h$ time units. We again calculate the route assignments based on these forecasts and implement them for the first $l$ time units, and so forth. The rolling-horizon approach can be used to keep updating the route assignments indefinitely.

In order to give a complete picture of the solution algorithm within the rolling-horizon framework, we refer to Figure 7.5. Here the algorithm with full information, described in Section 7.5.1 is referred to route assignment algorithm with full information. This algorithm will work as if the forecasts are the true information and generate the optimal route assignments for a length of $h$ time units with the given forecasts.

![Flow chart of the solution algorithm within the rolling-horizon framework.](image-url)

The only extra difficulty in this solution algorithm is to keep track of the paths and positions of the agents in the simulation. Ending the simulation does not mean that the planning horizon ends. In the next stages, the agents that are present at the end of the simulation still need route assignments toward their destination. Therefore, we keep track of the paths and positions of these agents and start the new simulation which includes this information.

### 7.6 Discussion

This section describes the correctness and boundedness of the simulation-based models. We also elaborate on how the simulation-based models can be used for the routing of containers in synchromodal networks.
Correctness and convergence  The only algorithm that needs a proof of correctness is the simulation-based algorithm from Section 7.5. The correctness of this algorithm relies heavily on the correctness of the algorithm from the paper by Peeta and Mahmassani [98]. The only difference between our algorithm and theirs is the use of different cost functions for the trains and barges. However, as in more literature on synchromodal systems, we assume there will always be a route possible via truck. The cost of this route will be finite and therefore, a shortest path will always exist. Assuming the correctness of the time-dependent shortest path algorithm, we can conclude the algorithm in Section 7.5 is correct.

Peeta and Mahmassani describe that because of the use of a simulation, there is no guarantee that there is a descent direction in each iteration. However, convergence of the algorithm was obtained in all experiments reported in [98] and many other test networks. As mentioned before, our algorithm only differs in the cost functions for the trains and barges. Therefore, our algorithm also convergences.

Boundedness  Some of the simulation-based models allow for rerouting, namely heuristic 2 and the simulation-based solution algorithm in the rolling-horizon framework, cycles may occur in certain routes. For some agents it may be beneficial to go back to a previous node on a later time step to shorten the total travel time. However, these cycles are bounded, since the total travel time needs to be diminished in order for an agent to take the new route. All lengths and costs are non-negative integers, therefore travel times can only be decreased until 0 in a finite number of steps.

Another part that may induce unboundedness is unbounded lengths and costs. For the travel time on roads, we know that a finite amount of agents is on the roads, resulting in finite travel times. For the trains and barges we do allow for infinite travel times. These infinite travel times arise if the last train/barge on a link has departed, see Section 7.3. But, as mentioned before, we always assume a container can travel by truck. This means there is always a possibility for a path with finite length/cost and this path will always be chosen by the shortest-path algorithm.

Use of the models  All these models attempt to route containers or vehicles at minimum cost. Nowadays, in intermodal networks, human planners make decisions on how to route certain containers. Although experienced, these planners usually have no guidance of quantitative methods. The simulation-based solution algorithm from Algorithm 7.5, may help these planners to make well-decisions.

All models mentioned in this chapter can also be used as reference for other methods of routing the containers. The simulation-based solution method may be optimal, but one needs to take into account details that arise in real-life implementation. If another method is preferred, these models may give a sense of how well that method performs.
Chapter 8

Results

In this chapter we apply the two heuristics and the simulation-based solution method to a synchromodal network. Section 8.1 describes this network and its parameters. Then in Section 8.2 we discuss how each model performs in this network and in some smaller examples. We also discuss the results of the analytic model and the solution algorithm in the rolling-horizon framework in Section 8.2.

8.1 Synchromodal network and parameters

We elaborate on the synchromodal network to which we apply our models. Section 8.1.1 describes the network, i.e. the nodes and the edges. Section 8.1.2 describes all the parameters such as arrival rate and capacities.

8.1.1 Trans-European Transport Network

The synchromodal network that we work with in this chapter is part of the Trans-European Transport (TENT) Network [148]. The network is very large and therefore we will zoom in on part of it, see Figure 8.1. We look at part of the North Sea - Mediterranean Corridor. We focus on the part of this corridor that crosses through The Netherlands, Belgium and a little in Northern France.
From the Trans-European Transport Network in Figure 8.1 we take into account the cities that are likely to have an intermodal hub: Amsterdam, Utrecht, Rotterdam, Breda, 's-Hertogenbosch, Bergen op Zoom, Nijmegen, Venlo, Middelburg, Antwerp, Maastricht, Ghent, Bruges, Brussels, Lille, Dunkirk, Calais and Namur. These cities are the nodes in our network. This means that in total we have 18 nodes. In Figure 8.1 we can see which nodes are connected to each other and by what mode.

Figure 8.1 shows exactly how each part of the network looks. This is very detailed, one can see how the rails and waterways run through the land. What our models need is only if two nodes are connected and if so, what the properties of this link are. We do not need to know how the waterway curves in between two nodes. Therefore, we simplified the network. This simplified network is shown in Figure 8.2. Here we only see the nodes under consideration and the links between nodes.
8.1.2 Parameters of the Trans-European Transport Network

In the previous section we described the network that we use to test our models. However, knowing only the network is not enough. We also need to know the properties of the edges; length and capacity. For the rail and waterway links we need to know the departure schedules too. And for all links, we need to know the costs of traversing this link. In addition to these properties we also need to know the rate at which orders arrive in the network from each node. Finally, we need to know the total planning horizon and, for the rolling-horizon approach, the rolling time. This section treats each of these parameters.

A note beforehand We want to note that there was no real data available as test input for our models. This means that all the parameters in this section are chosen by an educated guess. Not all parameters will be realistic, but this is due to the strong dependencies between certain parameters. For example, a capacity of 100 is a lot if only 20 containers need to be shipped within the planning horizon, but it is not nearly enough if 1000 containers want to traverse the link each minute. This means that for certain parameters we started at a certain value and adjusted all the dependent parameters accordingly. The implementation of our models can handle more realistic instances, where parameters change for each
train/barge that departs. However, choosing these parameters differently for each train/barge will only result in more freedom of choice.

**Properties of the edges** Each edge has a type, length, capacity and cost. The type of each link is already discussed in Section 8.1.1. Additionally, the rail and waterway links have a departure schedule.

- **Length:** the length of each edge \((u,v)\) is determined by looking at the amount of kilometres it takes to travel from node \(u\) to node \(v\). This length is equal for the road, rail and waterway. We note that it is possible in our implementation to change the length depending on the type of the link.

- **Capacity:** the capacity of a link has to match the amount of traffic that needs to be handled by the system. We fix the capacity and adjust the arrival rate accordingly. The capacity for each type of link is:
  - **Road:** capacity of 200 containers per link.
  - **Rail:** capacity of 20 containers per train.
  - **Waterway:** capacity of 100 containers per barge.

  We remark that these capacities are not entirely fair, since a shorter link has as much capacity as a long link. Certainly for roads, this is not true in real life. Choosing different capacities for each link however, would only result in more unknown parameters.

- **Departure schedule:** the departure schedule of the trains and barges is fixed and can be any given list of departure times for each individual link. However, we choose to standardise the schedules. The departure schedules are:
  - **Rail:** trains depart every 30 minutes, starting at \(t = 30\).
  - **Waterway:** barges depart every 120 minutes, starting at \(t = 120\).

- **Costs:** as mentioned before in Chapter 7, costs do not have to be equal to travel time. In a synchromodal network, this freedom is preferred, because if cost were to be equal to travel time, agents would never choose to use a train or barge. However, there are other costs to be taken into account, such as reliability and CO\(_2\) emissions. For the costs we worked with the following parameters:
  - **Road:** the costs of travelling a link are equal to the travel times.
  - **Rail:** the costs of going on a train are equal to the costs of a road at 25% capacity. This means that rail becomes interesting, when more and more containers are on the road.
  - **Barge:** the costs of going on a barge are equal to the costs of a road at 50% capacity.

When no traffic is in the system, the roads will be the cheapest option available. Since we are considering an agent-centric network, we think it is logical that containers prefer the road when all is quiet, since this will yield the fastest time. However, when a road becomes crowded, agents might prefer a train or barge, since this is a more reliable mode of transportation.

**Arrival rate** For the arrival rates, we divided the cities in three categories: small, medium large and large. A small city like Bergen op Zoom will generate a lot less traffic than a large city like Antwerp. The division into categories is as follows:

- **Small cities:** Breda, 's-Hertogenbosch, Bergen op Zoom, Nijmegen, Venlo, Middelburg, Maastricht, Dunkirk and Calais.
• Medium cities: Amsterdam, Utrecht, Ghent, Bruges, Namur
• Large cities: Rotterdam, Antwerp, Brussels and Lille

The arrival rates for small, medium and large cities are \( \frac{1}{10}, \frac{2}{5}, \) and 1 respectively. This means that small cities generate on average one order per 10 minutes, medium cities two orders per 5 minutes and large cities one arrival every minute.

**Time steps and planning horizon** The planning horizon is 8 hours, resembling a normal workday. The time step of the shortest-path algorithm is 5 minutes, thus we have a total of 96 steps. For the rolling-horizon approach we will use a rolling time of 12 time steps, i.e. 60 minutes.

### 8.2 Results of the models and routing methods

We developed two different models: an analytic model and a simulation-based model. On the simulation-based model we developed four different routing methods: two heuristics, a simulation-based solution algorithm and a model that incorporates the simulation-based solution algorithm within a rolling-horizon framework. The analytic model does not optimise the routing of the agents and therefore, notes on this model will be discussed separately in Section 8.2.3. The main part of this section is about three methods: heuristic 1, heuristic 2 and the simulation-based solution algorithm. First in Section 8.2.1 we elaborate on some scenarios that highlight differences between these methods. Then in Section 8.2.2, we show results for the TENT network described before. In Section 8.2.4 we elaborate on the solution algorithm within the rolling-horizon framework.

#### 8.2.1 Scenario analysis

In this section we elaborate on two small networks that are used to highlight the differences between the two heuristics and the simulation-based solution algorithm. These results will improve our understanding of each of these models.

We developed four examples in two small networks. The first network can be found in Figure 8.3 and the second network in Figure 8.4. Here one can see the length of the links and the capacity of each link (length/capacity). For each of these links the costs increase with the amount of agents in the network. One can understand that when a link is over-utilised, i.e. there are more agents on the link than the capacity allows, these costs increase steeply. When there is enough capacity for the amount of agents, the costs are similar to the free flow travel costs.

![Figure 8.3: Example 1.](image)
Figure 8.3 shows the first small network for scenario analysis, example 1. In this first example we want agents to travel from the left of this network to the right. This means they all first need to traverse a link that has a length of 10 and a capacity of 100. Afterwards, they can choose between one route with length 21 and capacity 10 or a route comprised of two links: one with length 10 and capacity 1 and one with length 10 and capacity 10.

In the second network, example 2, Figure 8.4, agents need to travel from the bottom to the top. There are two routes available, both comprised of two links. Both routes start with a link with length 50 and capacity 100. The first route then has a link with length 20 and capacity 100, while the other route has a link with length 5 and capacity 1.

In our examples we assume that 10 agents want to traverse the networks. Firstly, we assume that all these agents want to traverse the network at the same time. For example 1 these results can be found in Figure 8.5 and for example 2 in Figure 8.6.
In Figure 8.5, we can see that heuristic 1, i.e. the heuristic without rerouting, performs very poorly compared to heuristic 2 and the solution algorithm. The reason for this behaviour is that heuristic 1 looks at the network and sees the route via the link with capacity 1 as the shortest path. Since this path has length 30 and the other path has length 31. All agents depart at the same time, therefore no agents are on the link with the small capacity and thus the shortest route will seem to be the same for all agents. Therefore, all agents will travel via the lower route. However, when the agents do arrive at the second link in their route, all the other agents also want to traverse that route, which means there is an enormous extra cost for this link. Heuristic 2 does not show this behaviour, since the shortest path is recalculated at the first intermediate node. Here the first agent will travel the route with length 30, but the next agent sees that the link is already in use and will therefore use the link with length 21. This recalculation of the shortest path is not done in heuristic 1. The solution algorithm also divides the agents, such that they all have a low cost of travelling.

Figure 8.6: Costs for each agent in example 2 with clustered travelling.

Figure 8.6 shows the costs for example 2. Here both heuristics perform badly compared to the solution algorithm. The reason for this is that both heuristics see the route on the right side of the network as the shortest path at the departure time of the agents. Therefore, both heuristics send all agents on the right link with length 50. Heuristic 1 does not have a possibility to reroute and therefore all agents will be stuck on this route. This creates a situation where all agents want to traverse a route of capacity 1. This leads to extra costs. Heuristic 2 does have a possibility to reroute, but only at the intermediate node. This means that at this point agents would have to travel back in order to avoid the link with capacity 1. But this means extra costs of at least a link with length 50. Therefore, most agents will still be routed on the link with capacity 1. However, one can see that agents 8 and 9 have a lower cost with heuristic 2 than with heuristic 1. This means that at this point it is actually beneficial to go back to the first node and take the left route. The solution algorithm has full information and thus knows that sending all agents over the right route will lead to trouble. Therefore, the solution algorithm already divides the agents over the left and right route from the start.

These examples clearly show that the solution algorithm outperforms the heuristics in certain cases. However, this effect is most clearly seen when all agents depart at the same time. Figures 8.7 and 8.8 show the same examples, but 10 agents depart at 10 different time steps. This means that the first agent departs at time 0, the second at time 1, etc.
Both Figures 8.7 and 8.8 clearly show that in the case where the departures are spread out over the planning horizon, each model performs well. The spreading of the departures results in scenarios where all models are able to react to the fact that the link with capacity 1 is already used. Therefore, even with the heuristics, this is no longer the shortest path and thus agents will be spread out over the network. It is interesting to see in Figure 8.7 that the solution algorithm does not result in the minimum cost for all agents. This is more clearly depicted in Figure 8.9.

Figure 8.9 shows that for most agents heuristic 2 performs better than the solution algorithm. This is due to the time-dependent shortest-path algorithm used in the solution algorithm. As described in Chapter
7, this time-dependent shortest-path algorithm is discrete. That means that if the costs of travelling are not integral, errors are introduced due to rounding. In example 1 the free flow cost of travelling a link with length 10 is 1.5. That means that the rounding error is large and thus heuristic 2, which is continuous, can outperform the solution algorithm.

### 8.2.2 Results on the TENT network

In this section, we describe what the costs are under optimal routing on the TENT network. These routes are determined by the two heuristics and the solution algorithm with full information. The parameters of the network are described in Section 8.1.2.

Figure 8.10 shows a graph with the ordered costs for all agents in the network. One can see that there is almost no difference between the two heuristics and the simulation-based solution algorithm. This means that in our transportation network, there is little difference between taking the shortest route when only knowing the occupancy up to the time of departure, or knowing the occupancy over the entire planning horizon. The reason for this difference is likely the fact that the departures of the agents in this network are spread out. As shown in Section 8.2.1, spreading out the departures of the agents diminishes the difference between the different models.
Therefore, we also looked at the TENT network with peak moments. The results for this scenario are shown in Figure 8.11. Here the arrival rates are different than mentioned before to create peak moments. The parameters are as follows:

- In the first two hours, we want to create a large amount of traffic. The arrival rates for small, medium large and large cities are scaled up to $\frac{1}{5}$, $\frac{4}{5}$ and 2 respectively. The orders are also given different volumes, meaning that an order does not contain only one container, but might have more containers departing at the same time. The volume of the orders is a uniformly distributed number between 1 and 10.

- For the remainder of the planning horizon, i.e. the remaining six hours, we still want to have traffic, but not that much. In this time window the arrival rates for small, medium and large cities are $\frac{1}{200}$, $\frac{1}{50}$ and $\frac{1}{20}$ respectively. The volume for an order is still a uniformly distributed number between 1 and 10.
Figure 8.11: Costs of reaching destination node for each agent in the network with peak moments (ordered).

Figure 8.11 again shows little difference between the two heuristics and the solution algorithm. Our conjecture is that how the nodes are connected by the different links also influences the analysis of the models. In the two examples in Section 8.2.1, the networks are chosen specially to show differences. The TENT network has a lot of alternative paths and therefore it is easy to find an optimal route in this network, even when not all information is available.

8.2.3 Notes on the analytic model

Our analytic model does not provide a means to find optimal routes. Since the model is static, the routing probabilities need to be known beforehand. However, we can compare the number of containers on a link with both the analytic model and our simulation. If we set fixed arrival rates fixed and set the mean service time distributions in our analytic model to be the same as the mean travel times in the simulation, we can calculate the mean number of customers on each link analytically. In our simulation we can also keep track of the amount of customers on each link and therefore, we can compare these two models.

There is, however, an issue with the computational tractability of our analytic model. Calculating the probability that there is an exact amount of customers on a link is easy. The equations for this can be found in Chapter 6. Here \( \pi(n) \) is the probability that the state of the system is vector \( n \). This vector has a value for each queue and each customer class in the model. But to calculate the mean number of customers in the model we need to evaluate Equation (8.1).

\[
E[L] = \sum_{n \in \mathbb{N}^M \times K} \pi(n)
\]  

(8.1)
In Equation (8.1) \( n \) is a vector out of \( \mathbb{N}^{M \cdot K} \), where \( M \) is the number of queues and \( K \) is the number of customer classes.

Even if we would take a look at our example network from Chapter 6, we have 12 queues and 25 customer classes. Therefore, \( n \) is a vector with 300 values. This means that even if we would terminate the sum at \( n = (10, 10, \ldots, 10, 10) \), we still have to calculate \( 11^{300} \) possibilities. This is way too hard to calculate on a normal computer.

Since we are only interested in the mean queue length, we also looked at Mean Value Analysis. For a single-class network, mean value analysis is very fast and grows linearly with the number of customers and queues. However, in multi-class networks, mean value analysis grows exponentially in the number of customers. Practically, it only works for three to our customer classes. In our model, the amount of customer classes is the amount of nodes squared, which means we would only be able to analyse networks with one or two nodes with mean value analysis. Obviously, this would be too restrictive for the goals of this research. Unfortunately, this means that we have no results from the analytic model.

### 8.2.4 On the solution algorithm within the rolling-horizon framework

We were not able to implement the solution algorithm within the rolling-horizon framework due to time constraints. It would be interesting to see how the results differ between the algorithm that has full information, i.e. knows all orders beforehand, and the algorithm that uses forecasts.

If we were able to implement the rolling-horizon framework, we think that it again depends heavily on the parameters of the network how big the difference will be. As we have seen before on the TENT network, the differences are very minor. Therefore, our conjecture is that for this network working with the full information or stochastic information does not matter. On the other hand, if one would use forecasts on the smaller examples from Section 8.2.1, we might see differences. Think about a scenario in which the forecasts do not predict traffic on the links with small capacity, but when the container arrives, there is traffic on this link. Then we can see the same thing happening as with the heuristics and the simulation-based solution algorithm.

In conclusion, the reliability of the forecasts will have the largest impact on the performance of this model. Reliable forecasts give a similar performance to the simulation-based solution algorithm, while unreliable forecasts might be harmful to the costs. We do want to remark that in real life, this model will be preferred over the model with full information, since full information is never available ahead of time.
Chapter 9

Conclusions and discussion

This chapter briefly discusses the conclusions from our research.

9.1 Conclusions

In Part I we introduced synchromodal transportation and discussed the four quadrants that can be applied to synchromodal networks: limited, selfish, social and cooperative. The division is made between global and local information and global and local optimisation. We mentioned that in networks where information is shared, thus in selfish and social networks, this information might be public or private. This is shown in Figure 9.1.

![Enhanced framework, from [152].](image)

Figure 9.1: Enhanced framework, from [152].
Our research is focused on a selfish synchromodal network, which is also referred to as an agent-centric network. Here each agent wants to optimise its own optimisation objective. The information that is available in the network, is available for all agents.

Part II consists of an extensive literature review on synchromodal transportation. We missed a framework for this literature categorising the problems of synchromodal transportation in a compact way. This means that it is difficult to see the similarities and differences between papers dealing with synchromodal planning problems. Chapter 3 therefore presents such a framework. The literature review also shows us that literature on the operational, day-to-day planning of synchromodal systems is scarce. We aim to fill this gap in the literature by developing methods that might help a planner to schedule containers in a synchromodal network. These methods and their results are discussed in Part III.

We developed two different models:

- **Analytic model**: the first model is an analytic model based on stochastic networks. We modelled the agent-centric network as a *Multi-Class Open Product-Form Network*. Here each link in the network is represented with different queuing models, that aim to capture the relevant processes of containers moving over a road, rail, or waterway.

- **Simulation-based model**: the other model is an event driven simulation, that simulates the agents going through the synchromodal network.

For the simulation-based model we developed four different methods to generate routes for the containers. All methods aim to minimise certain costs. The four methods are described below:

- **Heuristic 1**: The first heuristic is a naive implementation with public information. We assume that each agent checks the public information at departure and will react accordingly. This means they will take the shortest path for the current state of the network. After the choice is made, they will not deviate.

- **Heuristic 2**: The second heuristic is very similar to the first, except the fact that agents do switch routes before reaching their destination. This means that at each decision point, i.e. an intermediate node in the network, they again check the state of the network and reroute if necessary.

- **Simulation-based solution method**: the simulation-based solution method assumes full information. This means that this algorithm assumes that each agent knows exactly what is going to happen for a certain planning horizon. This includes public as well as private information. With this information, the algorithm seeks to find the optimal routes for all agents that arrive somewhere in the planning horizon.

- **Simulation-based solution method in a rolling-horizon framework**: Private information will not always be accurate. Therefore, we also developed a model that assumes that for a short time period reliable estimates are known and for a longer time period estimates are known that are less reliable. This algorithm relies heavily on the simulation-based solution method.

The analytic model needs a lot of assumptions compared to a real life model. In a synchromodal network, we deal with a lot of dynamic aspects and stochasticity. The analytic model does provide choices to model the stochasticity, but it lacks the dynamic aspects. We developed a model that represents each link with two queuing models. We describe how the mean queue length of these queues can be found, knowing parameters like the arrival rate and the routing probabilities. However, the complexity of a synchromodal network makes the computation time for the mean queue length enormous. This means that we were
not able to draw conclusions from this model. For the simulation-based model we were able to develop four different methods to answer our research question: What is the optimal routing for containers in an agent-centric synchromodal system?

We are interested in the the effect of different types of information on the optimal routing. The research questions were:

- What is the optimal routing for containers in an agent-centric network with only public information?
- What is the optimal routing for containers in an agent-centric network with full information, both public and private?
- What is the optimal routing for containers in an agent-centric network with stochastic full information?

For the optimal routing with only public information we developed our two heuristics. Obviously, heuristic 2, that allows for rerouting, outperforms heuristic 1. The reason for this is that containers are able to change their route when new information enters the network. Heuristic 2 makes use of all public information that is available.

The optimal routing with full information can be determined with the simulation-based solution algorithm. An iterative process is used to find optimal routes for the entire planning horizon.

If the information about the future is stochastic in nature, one should use the simulation-based solution method in a rolling-horizon framework. The rolling-horizon framework adjusts the forecasts and works on stochastic information.

The two heuristics and the simulation-based solution method are tested on small examples and on a large synchromodal network, based on the Trans-European Transport Network. The results, presented in Chapter 8, show that the parameters and the network itself have a huge impact on the performance of the models. We mention the advantages and disadvantages of each model below. This immediately answers our last research question: What are the similarities and differences between the developed solution methods?

- **Heuristic 1**: this heuristic performs well if the arrivals of the orders are spread out over the planning horizon. It needs time to see the congestion of the network build up. This heuristic performs poorly on networks where orders need to travel a large distance. For all orders, the routes are determined at their departure time. That means that the more time it takes for an order to move through the network, the more can change in the network. This will result in sub-optimal routes.

- **Heuristic 2**: the second heuristic also performs better when the orders are spread out over the planning horizon. Although it does have the availability to reroute certain containers, the availability of alternative routes plays a huge role on its performance. If one of the routes seems like a short path at departure time, the other routes need to be reachable from intermediate nodes on that route. If there are no alternative routes available, the rerouting will not result in smaller costs. However, this model does not perform worse when the orders in the network need to travel longer.

- **Simulation-based solution method**: this last algorithm results in optimal routes for all discrete cases. This means that the parameters of the network and the network itself have less effect on this method. However, real life instances are continuous and the discrete nature of the simulation-based solution method is a disadvantage. Note that this problem can be solved by taking smaller time steps within the algorithm.
In terms of computation time we see that calculating the shortest paths is the part that takes longest for all three simulation-based models. Obviously, computation time increases when the number of nodes in the network is increased. The first heuristic calculates a shortest path for each order that departs and therefore the computation time also increases with the number of orders. The second heuristic calculates a shortest path for each order at the time of departure and at the time of arrival at an intermediate node. This means that the second heuristic is slower than the first heuristic. The computation time of heuristic 2 is dependent on the amount of orders in the network and the amount of intermediate nodes on a path. The simulation-based solution algorithm is less affected by the amount of orders that depart within the planning horizon. This algorithm has an iterative nature that switches between calculating the time-dependent shortest paths and a simulation of the containers travelling through the network. The shortest path algorithm will calculate a shortest path on each time step for all origin-destination pairs and therefore it does not depend on the amount of traffic that is in the network. In the analysis of the Trans-European Transport Network we saw that the heuristics take longer than the simulation-based solution algorithm to determine the routes.

In real life, full information will never be available and therefore, we discussed the rolling-horizon framework. The simulation-based solution algorithm can be put in this rolling-horizon framework to determine optimal paths under forecasts of future orders. Although this method is probably preferred in real life, we were unable to implement and test this method. The conjecture is that the reliability of the forecasts plays a great role in the performance of this method. Reliable forecasts give similar results to the simulation-based solution algorithm with full information. On the other hand, unreliable forecasts can result in sub-optimal routes.

In conclusion, we can see that additional information for the two small examples results in less costs for all agents, when all agents depart at the same time. This means that in certain instances the additional information in the network can be used to gain better performance. However, we have seen in the large Trans-European Transport Network, that the differences between the methods and thus between the usage of information, vanishes. This means that the additional information is only valuable for networks with certain structure and parameters. The networks need to have large differences between the links in terms of costs and capacity. In addition the orders should be bundled together, departing at times close together. Lastly, the networks cannot have a large amount of alternative routes that are similar to the optimal route. In a network in which all of this is the case, the additional information can really make a difference on the optimal routes for the containers in a synchromodal network. Tests on real-life networks are needed to determine the practical value of additional information in synchromodal networks.

### 9.2 Discussion and further work

We have discussed literature on synchromodal planning problems and developed five models that may help with the planning of container routes in a synchromodal network. Obviously, our research can be extended, this section elaborates on possibilities of further work.

#### 9.2.1 Results on real life networks

As discussed in Chapter 8, real life data was not available for this project. Since the network characteristics and the parameters of the network play a huge role in the performance of the different methods we are eager to see how the different methods perform on real life instances. The parameters needed to run each of the algorithms can be gathered from real life data. This way, one could compare the methods to each other and to the current planning of the containers. This can show how well the methods work compared to the actual planning of human planners.
9.2.2 Results on the analytic model

The way we tried to calculate the occupancy on the links in the networks was computation heavy. This means that results could not be gathered from the analytic model. However, closed formulas are known for the queue lengths of the different queues we used to model the synchromodal network. By using these closed formulas, expressions can be found for the mean queue length that are not computationally heavy to calculate.

9.2.3 Results on the rolling-horizon framework

The simulation-based solution algorithm can work well with forecasts and we thought about the effects of the stochastic nature of the private information. However, since we were unable to implement this method, we did not have results that back up these thoughts. Therefore, implementing and testing this method on the different examples might help to see the effect of the stochastic information.

9.2.4 System optimal routing

As discussed in Section 9.1, synchromodal transport networks can be divided in four quadrants. The selfish network has a lot of similarities to a social network. The main difference is that not individual objectives are minimised, but a global objective is considered. The models developed in this research can also be applied to other optimisation objectives and therefore can be easily extended to the case of system optimal routing. There might be cost reduction possible even for individual agents when a system optimal view is taken on the network. This is described by a paradox: Braess’s paradox.

Braess [24] discovered a paradox in traffic networks where adding a road to a road network can actually cause the overall performance measures to become worse. Figure 9.2 shows an example network in which the Braess paradox occurs when adding a road from A to B.

Figure 9.2: Braess’s Paradox

Figure 9.2 shows a simple origin-destination pair with two alternative routes: via A and via B. The dotted line between A and B is the road that we will add later. Suppose that 4000 drivers wish to travel from “START” to “END”. The travel time from “START” to B and from A to “END” is constant: 45 minutes. The other two roads have a travel time dependent on the amount of drivers, \( t = \frac{T}{100} \). If we look at the original situation, without road between A and B, we see that the travel time is either: \( \frac{T}{100} + 45 \)
or $\frac{T_B}{100} + 45$ with $T_A$ the amount of drivers that go via A and $T_B$ the amount of drivers that go via B. In equilibrium it is clear that $T_A = T_B = 2000$, thus the route from “START” to “END” will take $\frac{2000}{100} + 45 = 65$ minutes.

Now suppose the dotted line is a road that is available for all drivers. To traverse this road no time is needed. In this scenario a driver starting at “START” will always choose for the road: “START” to A. The worst-case travel time for this road is $\frac{4000}{100} = 40$ minutes, while the alternative road takes 45 minutes. Arriving in A the driver will switch to the road B to “END” by exactly the same reasoning. Thus it is an equilibrium, since no driver can improve by switching roads. However, the total time from “START” to “END” is now $40 + 40 = 80$ minutes.

Adding a road is thus not always beneficial for the system. This paradox is equivalent to the fact that the agent-centric case may be harmful to the system. We can use the same figure, Figure 9.2 to illustrate this. Consider the dotted line to be a road in this network, which takes no time to traverse. In a net-centric case the operator can choose to make half of the drivers go via A and half of the drivers via B and not use the dotted road. If every agent will choose their own fastest route, we arrive in the same situation as before. Here every agent will choose to go over the route: “START” - A - B - “END”. Thus looking at the same example from SO and UE viewpoint, even the marginal travel time per agent can be reduced when there is an operator.

Roughgarden and Tardos [115] quantify this degradation in network performance. They prove that the total travel time chosen by selfish agents, might be the same as the total travel time incurred by optimally routing twice as much traffic.

In conclusion, we think that including the possibility to switch from a user equilibrium view to a system optimal view might benefit the models greatly. However, the extension into a system optimal algorithm is not trivial and therefore beyond the of scope of this research.
Bibliography


[77] Li, L. “Coordinated Model Predictive Control of Synchromodal Freight Transport Systems”. In: (2016).


URL references


---

**Internal documentation and lecture notes**


