FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Using the Internet of Things and Real-time Data for Optimizing Freight Streams in Transportation Networks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Rafael Konlechner, BSc
Registration Number 1125679

to the Faculty of Informatics
at the TU Wien
Advisor: Dr.-Ing., B.Sc., Dipl.-Oec. Stefan Schulte

Vienna, 25th April, 2018

Rafael Konlechner	Stefan Schulte

Erklärung zur Verfassung der Arbeit

Rafael Konlechner, BSc
Straß 4, 2860 Kirchschlag

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. April 2018

Rafael Konlechner

Acknowledgements

I enjoyed the support of a great many people in the course of writing this thesis. First of all, I would like thank Stefan Schulte for supervising my thesis and giving me much needed guidance in scientific writing, showing a lot of patience and a thorough sense of quality. I would further like to express my gratitude to my superior, Alexander Marschoun, who provided me with the chance of writing this thesis, even allowing me to dedicate some of my working hours to that matter.

I would further like to acknowledge contributions from Iris Heckmann, Sebastian Fink, Christoph Hochreiner, Philipp Hönisch and Michael Pichler. Another special word of thank you goes out to my family, friends and to my girlfriend Julia, who continue to offer support and encouragement and went through times, where we did not see each other that often. Lastly, I would like to especially thank my colleague Richard Holzeis, who, besides the countless hours we spent coding on the weekends, provided me with lots of opportunities, ideas and was a constant source of motivation. Thank you!

Kurzfassung

Die effiziente Abwicklung von Teilladungstransporten stellt wegen kurzfristiger Änderungen im Transportbedarf für die strategische Fahrplanberechnung eine Herausforderung dar, die zu schlecht ausgelasteten Transportfahrzeugen und zu vielen ökonomisch und ökologisch problematischen Leerkilometern führt. In dieser Arbeit stellen wir eine technische Lösung zur Erfassung von Echtzeitdaten in Logistiknetzwerken vor, mit deren Hilfe gewisse Entscheidungen der Transportplanung und Fahrplanerstellung vom strategischen in den operationalen Horizont rücken, um die Effizienz von Teilladungstransporten zu erhöhen. Im Detail präsentieren wir eine Systemarchitektur, die es erlaubt Daten aus heterogenen Datenquellen — wie etwa Internet of Things (IoT) Sensoren und Datenservices — zu einem detailreichen, aktuellen Abbild der physischen Realität zusammenzuführen, um auf Basis dieses Abbildes mit Methoden aus dem Operations Research dynamische Fahrpläne abzuleiten, die besser auf Bedarfsfluktuationen reagieren können. Mit *Dynamic Transshipment with Time Constraints in a Finite Planning Horizon* und *Vehicle Assignment* definieren wir formal zwei mathematische Optimierungsmodelle, deren sequentielle Ausführung der dynamischen Berechnung von Fahrplänen dient und die Beschränkungen auf Lieferzeiten, Transportkapazitäten und Umschlagsvolumina berücksichtigen. In einer Auswertung mit operativen Daten eines österreichischen Logistikunternehmens zeigen wir die deutlichen Effizienzsteigerungen eines dynamisch errechneten Fahrplans, im Vergleich zu statischen Fahrplänen. Unsere Ergebnisse zeigen eine Verringerung der Gesamtzahl an gefahrenen Kilometern um durchschnittlich 15.2% und eine Erhöhung der Fahrzeugauslastung um 2.8 Prozentpunkte. Durch die Einführung eines Systems zur Behandlung von zu Verspätung führenden Ausnahmezuständen — etwa Fahrzeuggebrechen oder Verkehrsverzögerungen — zeigen wir zudem die Robustheit des Systems gegenüber ebensolchen Zwischenfällen.

Abstract

LTL (Less than Truckload) shipping has an efficiency problem, due to inflexible schedules and the inability to address short-term changes in demand. We propose a technical solution for improving the efficiency of LTL shipping by automating aspects of operational decision-making using rich, real-time information. In particular, we propose a framework for collecting heterogeneous data from Internet of Things (IoT) sensors and data services in the context of logistics networks to maintain a rich state of the environment as a basis for routing and scheduling decisions. We use this information for optimizing freight streams by applying standard Operations Research methods to the problem. In particular, we introduce a methodology for computing dynamic vehicle schedules based on dynamic demand in a level of detail that allows schedules to be planned and adapted in an autonomous decision process in day-to-day operations. To that end, we formally define *Dynamic Transshipment with Time Constraints in a Finite Planning Horizon* and *Vehicle Allocation*, two mathematical optimization problems that build a schedule based on dynamic demand, considering constraints on delivery times, capacities of vehicles, loading terminals and hubs. Finally, we demonstrate the performance of our approach with evaluations based on operational data from an Austrian logistics company. In these evaluations, we are able to reduce the total amount of driven kilometers by 15.2% and improve vehicle utilization by 2.8 percentage points. By introducing and testing a mechanism for incident handling, we also demonstrate the robustness of our framework against unforeseen delays, such as vehicle breakdowns or traffic congestions.

Contents

Contents **11**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 4
1.3 Aim of the Work . . . 5
1.4 Methodology and Approach . . . 6
1.5 Structure of the Thesis . . . 6

2 Background and Foundations **7**
2.1 Less than Truckload Shipping . . . 7
2.2 *Industry 4.0* . . . 9
2.3 The Internet of Things in Logistics . . . 10
2.4 Radio Frequency Identification . . . 10
2.5 Complex Event Processing . . . 11
2.6 Decision Support Systems . . . 13
2.7 Cloud Computing . . . 13
2.8 Operations Research . . . 15

3 Related Work **17**
3.1 The Internet of Things for Real-time Data Capturing in Logistics . . . 17
3.2 Event-driven Architectures and Event Processing . . . 18
3.3 Autonomous Control and Decision Support . . . 19
3.4 The Transshipment Problem . . . 20
3.5 FZI ProveIT . . . 22

4 Methodology **23**
4.1 Properties of the Considered Transportation Network . . . 23
4.2 Event Load Estimation . . . 24
4.3 Architectural Guidelines . . . 25
4.4 Software Agents . . . 27

5 A Framework for Real-time Optimization of Freight Streams **29**
5.1 Architectural Overview . . . 29

5.2 Data Model . . . 30
5.3 Event Model . . . 33
5.4 Event Processing . . . 34
5.5 A Software Agent for Dynamic Vehicle Routing . . . 38
5.6 Instruction Model . . . 41
5.7 Future State Prediction . . . 43
5.8 Subsystem for Unexpected Event Handling . . . 44

6 Dynamic Transshipment with Time Constraints 49
6.1 Overview . . . 49
6.2 Dynamic Transshipment with Time Constraints in a Finite Planning Horizon 50
6.3 Vehicle Assignment . . . 55
6.4 Conclusion . . . 58

7 Evaluation 59
7.1 Performance of Routing Policy . . . 59
7.2 Performance of Incident Handling . . . 67
7.3 Weaknesses and Open Issues . . . 70
7.4 Evaluation Summary . . . 71

8 Conclusions 73
8.1 Summary . . . 73
8.2 Future Work . . . 74
8.3 Closing Remarks . . . 75

Acronyms 77

Bibliography 79

Appendix 88

CHAPTER 1

Introduction

1.1 Motivation

New technologies are reshaping the logistics landscape: Autonomous driving will soon reach a stage of industrial application and manufacturers continue to move towards the so-called Industry 4.0, allowing them to produce customized mass products on demand by connecting machines, workpieces and systems with Internet of Things (IoT) technology to form intelligent, autonomous networks of cyber-physical systems [NOV15]. This shift has and will have a profound impact in the logistics and transportation service industry, leaving logistics providers with major challenges in meeting modern demands [LSD+15]. Smart transportation systems and the IoT will play a key role in developing efficient and sustainable logistics and will embrace trends and developments in manufacturing.

1.1.1 State of Logistics

Logistics is one of the most dynamic sectors of the European economy, contributing to economic growth and international competitiveness. The sector accounts for a share of 7% of GDP in Europe and an overall freight volume of 2200 billion tonne-kilometers [Eur16]. Road transport takes the biggest share, at about 75% of all transported goods in the EU. It connects regions of high industrial activity with more remote areas over the European road infrastructure, creating an integrated transport network and thus enabling a single European market. Figure 1.1 provides an overview of the modal split of European member states.

The freight volume is growing steadily. By 2050, the overall volume of transported goods in the EU is projected to triple compared to the volume shipped in 2000. As freight volumes are growing, so is congestion. Drivers are confronted with increasingly congested roads while one out of four heavy goods vehicles still runs empty [ec211]. Figure 1.2 provides an overview of the share of all empty runs by European member

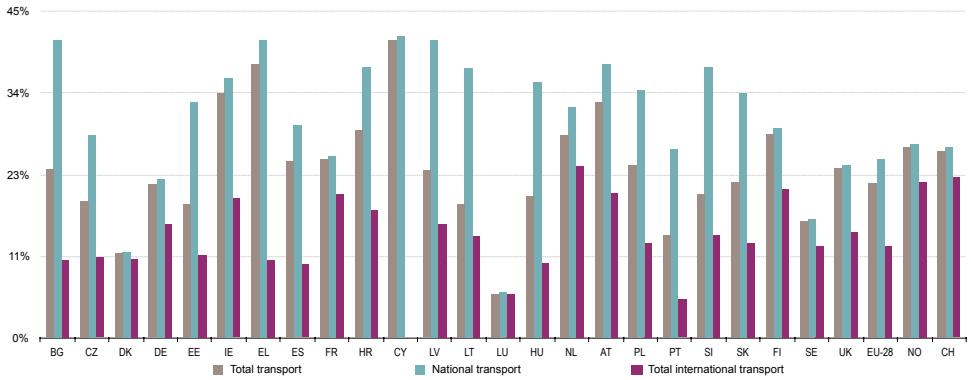


Figure 1.1: Modal split of domestic freight transport per country, 2013 (% of total tkm) [Eur13]

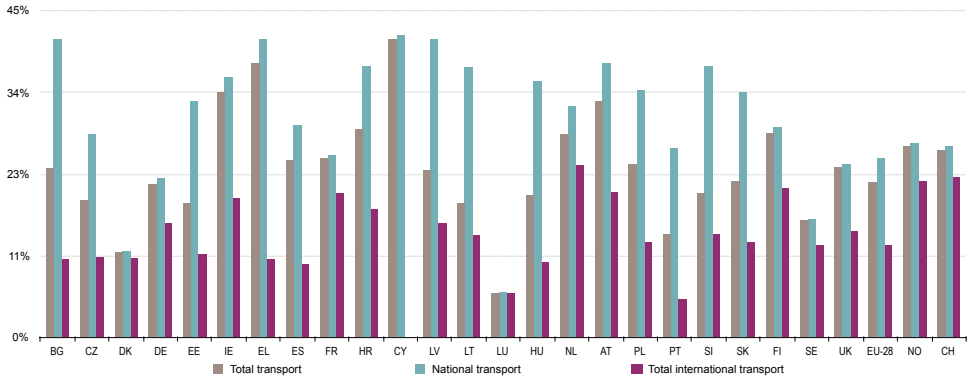


Figure 1.2: Share of empty journeys by type of operation, 2014 (% in vehicle kilometers) [Eur14]

states. In the face of these prospects, the objective is to build smart, well-utilized and safe road transportation systems, that ensure economic competitiveness and minimize environmental impact.

For the last 15 - 20 years, research in logistics and supply chain management has been focused around the development of tracking technologies such as Radio Frequency Identification (RFID) or the Electronic Product Code (EPC) standard, that can continuously track the state of physical objects such as cargo, trucks or loading equipment across the entire supply chain [SADP10, MD16, DHS07, SAB07]. Combined with other sensors (e.g. GPS), these real-time tracking technologies can be used to maintain a digital copy of the physical environment, which forms the base of smart transportation systems [MGYA14].

Despite many efforts, there is not yet one single standard for information exchange on

the state of logistics objects. Many of the proposed technologies and standards, including EPC have shown to be slow in adoption by the industry [NOV15]. This phenomenon has a couple of reasons. Investing in new technologies and betting on new standards is always associated with high costs and a certain risk. New standards are not immediately beneficial for a migrating business - not until they become widely used. It is of course hard to predict, which new standard will establish itself. For these reasons, logistics providers still heavily rely on outdated and unsuited software to a large extent. Those systems do not leverage modern information technology, which would yield big optimization potentials. In addition, integrations for third party systems are costly and lead to long-term commitments with business partners. This limits short-term flexibility and the possibility of ad-hoc partnerships. When building a transportation system, these issues have to be considered. Ideally, a modern transportation system can cope with heterogeneous data from a wide range of data sources and integrates easily with software from business partners [Gia09].

1.1.2 Future Trends in Logistics

According to a number of trend reports [Kü13, Kü14, MBC15, Yee15], the logistics sector will go through some key business and technology transformations within the next years. Mayor trends include:

Real-time tracking Using low cost sensors, the IoT and cloud computing technology, rich information about the state of shipments, the transportation fleet and traffic (e.g. temperature, humidity, acceleration, parcel dimensions, GPS) can be collected in real-time [Kü14]. This technology serves as a prerequisite for other trends, such as anticipatory logistics and shareconomies, described later in this section.

Omni channel logistics Modern retail happens on many channels. A customer might shop for a product in the store and later buy it online in a customized variation. Logistics providers have to make strategic decisions about warehouse placement and need to provide flexible logistics services for their B2B customers. This again is only possible, when maintaining rich information infrastructure, reaching across many business partners [Yee15].

Anticipatory logistics "Data is the new oil"[1] Real-time tracking and big data technologies enable logistics providers to collect vast amounts of data and set anticipatory actions based on insights gained from the data. This allows for better demand prediction and yields better network utilization, higher efficiency and service quality as well as faster delivery times. Anticipatory actions include anticipatory capacity planning (e.g. parcel volume prediction), anticipatory shipping and anticipatory risk management [MBC15].

[1] http://ana.blogs.com/maestros/2006/11/data_is_the_new.html

Shareconomies Sharing logistics infrastructure and crowd-sourcing certain aspects of the supply chain (e.g. delivery) can have manifold benefits, including improved elasticity of resources for changing demand and better utilization, new hybrid business models and reduction of CO^2 emissions [Kü14].

Autonomous Vehicles Self-driving vehicles are expected to have the most noticeable impact in the logistics industry within the next 10 years [Kü16], yielding cost reductions up to 75% and an increase in productivity of up to 25%[2], making a strong case for fast adoption of self-driving vehicles in the logistics industry. Autonomous technologies have been successfully deployed in closed environments, e.g. in warehouse operations and in some cases allow for semi-autonomous driving on highway sections in line haul transportation, but soon will reach a level of maturity that will enable the application of autonomous driving technologies throughout all logistical responsibilities: from warehousing to line haul and last-mile delivery.

1.2 Problem Statement

While containers and loading equipment have been standardized, information exchange about the state of shipping goods is still lacking the same degree of standardization, leading to the fact that many Information and Communications Technology (ICT) systems in the transportation industry are proprietary and only work within the boundaries of a company [Gia09]. As a result, real-time data covering cross-organizational logistics processes is not available. This prohibits predictions and anticipatory actions, whether autonomous or human, that would lead to better utilization of transportation networks and overall less empty kilometers driven. Instead, changes in volume or other unforseen events have to be handled manually [DDKS15], done by so-called schedulers. Schedulers decide, how to distribute cargo to available trucks to get it to its destination. Those trucks are often routed based on a fixed schedule, that determines which truck leaves for which destination at what time, making reactions to changes in volume or other unforseen events very unflexible. Schedulers decide mostly based on experience and expert intuition, as they do not have rich data about the current state of their or their business partner's transportation network [Gud07]. This might lead to situations, where the decision was appropriate from a local perspective, but would have been decided otherwise with more information available.

Controlling freight streams with scheduling and routing can be vastly improved by building an integrated framework for real-time data, that is able to collect data from different data sources (e.g. IoT devices and sensor networks) and use it to make well informed decisions. This allows decisions to be made autonomously and the detection of anomalies and subsequent reactions to be made much faster [NLW+12, PGM12].

[2] https://www.flexport.com/blog/self-driving-truck-automation-of-million-jobs/

1.3 Aim of the Work

While the technological foundations for real-time tracking, i.e. IoT and cloud computing technologies, such as Complex Event Processing (CEP) have been established [EN10], there has been little effort in creating a high-level integrated framework for road transport logistics [Gia09]. Such a framework would steer freight streams to avoid empty runs, schedule hauls in a way that all customers receive their shipments on time and quickly react to changes in the schedule, e.g. because of a technical difficulty with a truck. The framework would collect data from vehicle and warehouse sensors, cargo tags, transport orders from business partners as well as traffic and weather services for making well-informed, anticipatory routing and scheduling decisions.

This work aims at providing such a framework. It uses data from cargo tags (e.g. RFID), vehicle and warehouse sensors (GPS, utilization), transport orders as well as traffic and weather information services in order to estimate future network utilization and to process incoming transport orders in a way that optimizes the overall driven distance, avoiding empty runs. For this purpose, a software agent will be provided. This agent makes autonomous decisions as reactions to changes in volume based on a mathematical optimization model. In more detail, the presented concepts are:

Architectural Model The first step of the work addresses the subject of data collection and data representation. It discloses decisions about what data is used, how that data is represented in a data model and how the data is used for optimization. In other words, it defines the mapping of physical objects to their digital counterparts. These modeling decisions impact the software agent's view on the environment and influence the decision making process [Sch12, JBWL06]. To that end, an event-based data model that allows the access of a complete, high-level view of a road transportation network by means of snapshots will be introduced. A snapshot joins the latest available information about every object in a transportation network (i.e. each truck, warehouse and cargo that is labeled with a sensor or passive identification chip). The collected event data includes entry and exit scan information for cargo entering or leaving a hub, the current geo position of trucks, warehouse sensors that measure current warehouse utilization as well as a traffic and a weather service.

Optimization Agent In a second step, the possibilities of anticipatory scheduling will be demonstrated by proposing a decision agent, based on mathematical optimization with IBM CPLEX. This model will use the collected data to schedule cargo and route the vehicle fleet dynamically, minimizing the kilometers driven. The model will consider changing travel times caused by unexpected delays (e.g. traffic congestion) and delivery date constraints, since non-adherence may not only impact customer satisfaction, but also lead to penalty costs.

1.4 Methodology and Approach

A number of steps are necessary to provide a reasonable architecture and optimization model:

1. A survey of related work on *Industry 4.0* [WWB15, LSD+15, Dav15], industrial use of IoT [AIM10, DXHL14], CEP [EN10] and Operations Research (OR) in logistics [DDKS15, Gud07, EN10] will be conducted. This will disclose assumptions about the shape and functionality of the proposed framework as well as embedding the work into the bigger context of modern day logistics.

2. A framework for high-level, heterogenous event data will be introduced, following the guidelines proposed in [CDCMB14, CBM+13].

3. A separate simulation application will be built, capable of simulating all events considered by the framework and simulating work load on the transportation network based on a real transportation data set, as well as simulating various anomalies (e.g. leading to unexpected delays) [MKRW10].

4. An optimization model will be built, exploring both exact and heuristic solving methods for different problem sizes. In order to build this optimization model, reasonable assumptions about costs, utilization calculation and shipment parameters will be necessary (e.g. defining the utilization of a warehouse at a given point in time) [DDKS15, Pin15].

5. The proposed optimization approach will be evaluated in terms of a cost analysis, taking into account a realistic set of transport data [MKRW10]

6. The results will be compared with a baseline, that is set by a simple, static timetable routing algorithm, commonly used in the logistics industry [Gud07].

1.5 Structure of the Thesis

The reminder of this thesis structures as follows: Chapter 2 will provide the necessary foundations and terminology for subsequent chapters. Chapter 3 will present a literature study and related work regarding the fields of Decision Support Systems (DSSs) and applied IoT in logistics. The chapters 5 and 6 are dedicated to the elaboration of the first and second part of the solution respectively. The first part will cover aspects of the proposed framework, namely the data collection and representation, as well as the software agent interface. The second part will provide definitions for the mathematical models that are used to build the dynamic vehicle schedule. The evaluation, based on simulated scenarios, will be presented in Chapter 7. To conclude the work, Chapter 8 will provide a summary, closing statements and future prospects.

CHAPTER 2

Background and Foundations

This chapter will provide an overview of all necessary concepts and terms used in this thesis. First, a short introduction into the application domain of road transportation will be provided to help disclose assumptions about the shape of the transportation network that is considered in this thesis. Subsequently, the main technological prerequisites for this work will be introduced. The chapter concludes with a simple and practical example for mathematical optimization: The Transshipment Problem.

2.1 Less than Truckload Shipping

Road transportation can be classified into two different categories. If the cargo occupies a whole trailer, the term truckload or Full truckload (FTL) shipping applies. This type of shipping is often done by direct transport from the consignor to the consignee. In this case, there is no need for hubs or warehouses to be owned by the logistics provider. Thus, direct transports are often done by trucking companies, that only maintain a truck fleet [Gud13]. Unlike container shipping or FTL, where an entire trailer load is contracted to a single customer, in Less than truckload (LTL) shipping, the cargo of one customer is not large enough to occupy a whole truck. For LTL shipping, schedulers pool several orders onto one truck to use its capacity as economical as possible [Gud13, Chr16]. A typical scenario for the steps in the lifecycle of a shipment are depicted in Figure 2.1. The three mayor steps include collection, line-haul and delivery. Traditionally line haul connections, i.e. transport between hubs, were scheduled according to fixed time tables. These time tables are updated very infrequently (yearly, quarterly or monthly update cycles might apply [Gud07]) and do not consider changes in short-term demand. This can be subject to optimization. In a hub system, where shipments can pass several intermediate hubs before reaching its destination, a dynamic rerouting of shipments to different intermediate hubs might lead to better utilization of resources. The purpose of

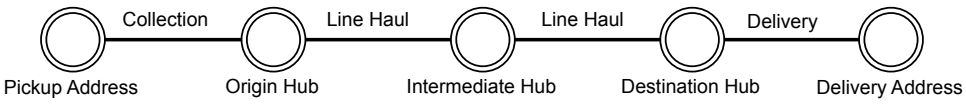


Figure 2.1: Typical shipping process for LTL shipping in a hub network

the optimization agent presented in this thesis is to find optimal combinations of LTL orders and truck schedules that optimize for transportation costs and avoid empty runs.

Note: The topics of collection and delivery also offer large optimization potential, but are out of the scope of this work, as they require fundamentally different optimization approaches that are subject to a number of research projects [GVdVV11].

2.1.1 Typical Shipping Process

In the typical scenario covered by the thesis a customer orders a transportation from one place to another, specifying the weight and volume of the cargo. The cargo is then picked up at the specified pickup time and brought to the nearest hub by local transportation. The scheduler then decides, which transport routes the cargo will be taking to reach the hub that is closest to the destination address. From the destination hub, the cargo is finally delivered by local transportation.

Transport Orders

Transport Orders are usually issued days before the desired pickup time, but no later than a day before. Some customers, e.g. manufacturers that want their products to be picked up periodically, use standing orders that arrange pickups in periodic intervals. This makes anticipatory planning much easier, even though sometimes customers do not specify the exact weight and volume of the cargo[1].

2.1.2 Logistics Decision Making

Decision making in logistics considers different horizons of time, depending on the lastingness and severity of effects of the decision being made [Gud13]:

- *Planning*: Planning describes long-term decisions concerning the design, organization, dimensioning and optimization of logistical processes, networks and resources in order to fulfill transport orders and adhere to quality requirements. Planning is often done based on fuzzy data, forecasts and uncertain expectations. Decisions

[1] This information was acquired in a series of interviews and personal experience during my work at DB Schenker AG in Vienna.

with consequences reaching for years are called strategic (e.g. building a new warehouse), whereas consequences reaching for months are called tactical (e.g. creating a new timetable for truck departures).

- *Scheduling*: Scheduling is concerned with the allocation of resources in order to fulfill transportation orders under certain constraints. Scheduling is based on actual customer orders, predictions from the planning phase or short-term demand forecasts, but is in any case much more certain knowledge than data from the planning phase. The operational scope for scheduling reaches from a couple of hours to a couple of days and is also known as Supply Chain Event Management.
- *Controlling*: This term bundles all controlling and interference aspects of actual operations (e.g. rerouting a truck on the road, loading, unloading and shipping of goods) and the execution of internal orders. In controlling, all details such as quantities, shipping contents, and deadlines are fixed as opposed to planning and scheduling, where data can be based on predictions.

The proposed optimization framework will be able to set instructions a few days ahead, e.g. for routing trucks. For this reason, it will have to work with predicted states, e.g. to determine where the trucks will be at a certain time in the future, assuming they arrive according to the schedule. Thus the framework will make decisions in the scheduling horizon. It will however also be able to make controlling decisions, if unexpected delays were to happen, that would make the planned schedule infeasible.

The framework could also leverage short-term demand forecasts to make better routing decisions [Mei11]. These forecasts could use historic data in different scopes, e.g. transport volume of the last days, transport volume of the last years at that time as well as information about holidays or special events. Demand forecasts are not within the scope of the thesis, but predictions about the future state of the transportation network, including predictions about utilization of trucks and hubs will be covered.

A number of ICTs and frameworks exist for distinct processes of logistical decision making, which will be covered later in the thesis. Section 2.6 introduces the concepts of such tools and Section 2.8 will offer a short introduction to OR, the scientific discipline of optimal decision making based on mathematical optimization.

2.2 *Industry 4.0*

Thoughts about the future of logistics information systems cannot happen without considering future trends in manufacturing. *Industry 4.0* stands for a digital transformation process, that industries are experiencing at the moment. It is considered the next step in the evolution of industrial production towards intelligent networks of cyber-physical production systems that can control each other autonomously. By connecting machines, facilities and systems and by automatizing modern manufacturing, efficient mass production of customized products will be possible. These advances will have profound impacts

on logistics. Machines will be able to autonomously communicate with suppliers for ordering the necessary resources which could then be automatically forwarded to logistics providers for performing the transport. This will require very flexible logistics, being able to process new orders with much shorter lead times and modern communication that can easily integrate with a wide array of business partners.

2.3 The Internet of Things in Logistics

The development of *Industry 4.0* heavily relies on IoT technology. IoT can be defined as a technological infrastructure, that extends internet connections to physical objects that are not computers in the classical sense, i.e. sensors, actuators, tagged objects, embedded systems, and mobile devices [VKAB+11]. Another common definition defines the IoT as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual objects have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network [Kra08].

IoT related technology is expected to transform a large portion of operations and processes in manufacturing, logistics and other industries [DXHL14, VKAB+11]. Estimations predict that by 2025, over 50 billion devices will be connected to the internet [SGFW10]. In a logistical context, IoT could be used to create anticipatory transportation systems, tracking each vehicle's location and monitor shipment movements and conditions by deploying a variety of different sensors and scanning devices. Many of these devices already exist today, but most systems use them in an isolated environment and for very specific purposes instead of using them in an integrated, generally available way for day-to-day decision making [CGP09, Gia09].

2.4 Radio Frequency Identification

The term IoT was initially proposed to refer to RFID[2], the first and most widely adopted IoT technology. With RFID, physical objects can be uniquely identified and tracked by tagging them with electromagnetic RFID tags and using a system of scanners and middleware to track the objects' state [SADP10]. RFID has been successfully applied to a wide range of areas in different industries, ranging from manufacturing, retailing, shipping, port operations, agriculture, pharmaceutical packaging processes, etc. and serves as an enabling technology and prerequisite for more automated and efficient business processes [MD16]. However, it has been argued that relatively high costs are still a major concern for businesses that consider a migration from barcodes to RFID [WNLY06].

[2] http://www.rfidjournal.com/articles/view?4986

2.4.1 Electronic Product Code and *EPCglobal*

The *EPCglobal* network is the most widely adopted industry standard for RFID-based information infrastructures. It uses the EPC to globally identify items based on a unique identification number and provides a system of RFID transponders and readers for identification, middleware, and an information and discovery service, as well as an object name lookup service, similar to the Domain Name System (DNS) [SAB07]. The goal of *EPCglobal* is to achieve real-time visibility of items throughout the supply chain, from the manufacturer to the retailer. EPC is designed to also work with many existing coding and tagging schemes, including barcodes. The canonical representation of an EPC code is a Uniform Resource Identifier (URI) [MGYA14].

In a typical scenario a manufacturer would register each transponder (i.e. tag), uniquely identified by the EPC, in the global object name service. By publishing the object in the lookup service, the digital trail of the physical object becomes accessible by all partner nodes within the supply chain and all information regarding the items are shared over the EPC information service in real time.

2.4.2 Embedded Devices

Electromagnetic scanning techniques are actively used in daily operations across many areas in the supply chain, but they have their technological foundations in the 1980s [MD16] and bring certain limitations. The biggest limitation is the lack of visibility in between scans [MGYA14]. Updates on the state of tagged items can only occur when the object is scanned, not in between scans, since the tags themselves are not connected to the internet. In many cases, this is no longer suitable for modern day logistics and leaves room for improvement [DHS07]. For example modern, event-driven transportation systems and real-time anomaly handling need these updates much more rapidly in order to maintain an accurate digital copy of the state of vehicles and shipments. Thus, advances in the low-cost production of embedded computing devices and networking technologies gave incentive for new research on smarter identification devices that directly communicate with central information systems and form a network of cyber-physical systems and sensors that can be monitored in real-time [MGYA14]. These advances are considered state-of-the-art in IoT in logistics and will be covered in Chapter 3.

2.5 Complex Event Processing

Since the proposed framework uses event-based, real-time information, one major technological prerequisite is CEP. The high number of incoming events necessary for maintaining a rich and informative digital image of the observed reality requires a systematic, fast and secure processing of events [EN10, PGM12]. CEP bundles methods, techniques and tools for processing such events while they are happening, meaning continuously and near real-time. By leveraging CEP, higher-level knowledge can be derived by accumulating information of primitive events or raw measurements over time [EB09]. For example,

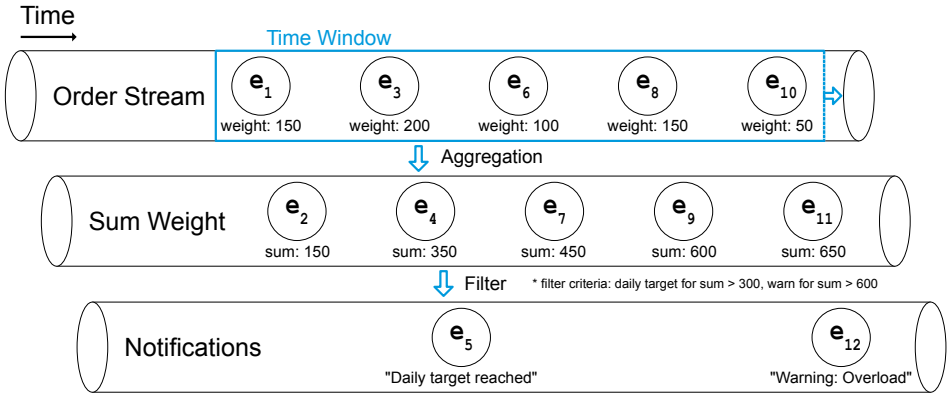


Figure 2.2: Aggregation and Filtering in Data Stream Processing.

the proposed framework can detect incidents by listening to GPS position events. If the latitude and longitude values of an observed vehicle do not change for an unusually long period of time, a warning is displayed and certain countermeasures could be initiated. In addition, other sensors might report information about mechanical failures (e.g. high engine temperature).

2.5.1 Data Stream Processing

Data stream processing tackles the problem of deriving higher-level knowledge from simple events. This is done by selecting, aggregating, joining and operating on (operations similar to SQL) incoming data streams to produce new higher-level data streams as outputs. Data Stream Management Systems (DSMSs) have their foundations in Database Management Systems (DBMSs), but differ in the rate of data changes. Updates in databases happen relatively infrequently in comparison to the transient, continuously changing data in data streams. Therefore DSMSs execute their queries or rules continuously, while queries in DBMSs are only executed once [CM12]. Figure 2.2 shows an example of stream processing. This example application generates events, if the aggregated volume of incoming transport orders reaches certain, predefined thresholds. The proposed framework leverages Data Stream Processing to collect freight volume information from transport orders in order to predict peaks of high or low utilization, which can be used to adapt resources accordingly (e.g. acquire additional hauliers for a strong week). If a new transport order is issued, the weight, volume and the pickup time are specified. By aggregating all incoming transport order volumes on a timeline by pickup time, the overall transport volume for any given instance in the near future can be predicted.

2.6 Decision Support Systems

A DSS is a specific information system that supports the process of problem solving and decision making [PGM12]. Properly designed, a DSS compiles and filters relevant information from sensor and event data, documents, knowledge databases and personal knowledge to assist in choosing the right alternative. DSSs are classified in (1) communications-driven, (2) data-driven, (3) document-driven, (4) knowledge-driven, and (5) model-driven systems.

For a practical DSS in a dynamic setting, as it is given by the problem covered in the thesis, three desirable characteristics can be identified [PGM12]:

- **Event-driven**: Decision support should be based on accurate data and get continuous updates about the changing environment.
- **Parallelized**: Many of the supported problems require prompt answers (e.g. accident handling in logistics [NLW+12]). Thus, underlaying optimization algorithms should be executed in parallel, taking advantage of distributed computing architectures [CGP09].
- **Flexible**: Processes and procedures within a company change over time and the landscape of decision problems is vast [PGM12]. Thus, DSSs should be easily extensible to account for specific aspects of different applications and should easily be adaptable to a continuously evolving environment.

As stated in the previous chapter, the proposed scheduling and routing framework will be able to operate autonomously. It does however operate very similar to DSSs, which is why the presented guidelines also apply for the framework. In fact, the framework could be considered a DSS, if all instructions set by the framework were sent to a human operator that could potentially interfere, instead of directly sending the instructions to the (embedded) devices for execution.

2.7 Cloud Computing

Cloud Computing is a computing paradigm where computational and infrastructure services (e.g. servers, storage, applications, services) are provided as a utility, over a network [SHG+10]. Sharing those resources enables on demand and pay-per-use provisioning with little to no overhead or prior commitment. By hosting software solutions at third-party cloud providers, companies avoid acquisition and maintenance costs of their own infrastructure and can dynamically scale their software solution stack, depending on the current work load. Cloud computing can be provisioned by means of reservation or on-demand. While provisioning by reservation is usually less expensive than on-demand provisioning, there exists the risk of over-provisioning resources which causes unnecessary costs. Cloud computing services are classified by different service layers, depending on the level of abstraction and underlaying services:

Infrastructure as a Service (IaaS) is a low-level computing service, where the provider hosts a virtual machine placement. Usually there exists a wide range of different virtual machine configurations with different storage, memory and CPU options[3] to suit the needs of many different purposes. Examples include Amazon EC2 instances and Microsoft Azure.

Platform as a Service (PaaS) provides a development platform with a set of services to assist the design, development, deployment and monitoring of software in the cloud, as well as offering the orchestration of additional software services (location, storage, authentication, etc.). Examples include IBM Bluemix, Google App Engine and Heroku.

Software as a Service (SaaS) is software intended for end users and hosted in the cloud. Instead of manually installing software, SaaS applications are usually consumed via a web browser, saving the trouble of installing and updating the software on a device and preventing compatibility issues. Examples include Google Docs, Amazon and Facebook.

2.7.1 Cloud Computing in Logistics

Cloud computing offers five essential characteristics that are of special interest in the application domain of IoT and logistics [KK16]:

On Demand Service Resources can be allocated and released automatically, without any need of human interaction with the service provider. This allows self-adapting systems to request resources on demand according to the current workload.

Broad Network Access Computing and infrastructure services are provided by means of standard internet protocols and mechanisms. This allows for a wide range of supported devices (PCs, phones, embedded devices).

Resource Pooling Cloud Computing service providers share and dynamically allocate their resources with multiple customers, which increases efficiency and utilization.

Rapid Elasticity Cloud computing allows for a very high degree of scalability, by dynamically allocating and releasing resources. In theory, this gives access to an infinite amount of computational resources from anywhere.

Measured Service Cloud computing providers measure the resource consumption of computing and infrastructure use and invoice based on a pay-per-use billing model.

Fields of application for Cloud Computing in logistics exist in cloud based DSSs [NLW+12] and IoT information systems [HZLQ15] (both will be covered in more detail in Chapter 3) and autonomous logistics [SHG+10]. The goal of autonomous logistics is to develop

[3] https://aws.amazon.com/ec2/pricing/on-demand/

transportation systems that require minimal human interference. Many of the proposed solutions for autonomous transportation systems choose a distributed, multi-agent approach [Sch12, JBWL06]. This way, the high overall complexity of logistics processes and decision problems is handled in a decentralized manner and can be done in parallel, which vastly improves scalability.

The proposed framework takes a centralized approach for an autonomous transportation system. This way, the software agent (i.e. the decision unit for making scheduling instructions) can use the state of the whole transportation network for scheduling and routing decisions, rather than only having a local view. The elasticity of cloud computing infrastructure and easy data synchronization make cloud computing a suitable paradigm for this centralized approach.

2.8 Operations Research

OR is the scientific discipline that provides the formal foundations and analytical methods to solve decision problems by means of mathematical modeling. In a wider sense OR also includes tasks regarding the problem definition and problem description, data acquisition and data analysis, which bridge the gap to other scientific disciplines like Management Science and Data Analytics [WN13]. Primarily, and in a more narrow sense, OR is defined as the mathematical modeling of decision problems as well as the development of algorithms to solve those problems efficiently [DDKS15].

Planning (i.e. decision making) which also applies to OR supported planning, is a well-structured process that follows six distinct steps [DDKS15]:

1. **Problem Analysis:** The decision problem and possible actions are formally defined.

2. **Goal Definition:** Rational acting needs to have a goal. Since never all possible aspects of optimality can be considered, this simplifies the model of the real world.

3. **Mathematical Model Creation:** The problem is formalized by defining decision variables and constraints on those variables.

4. **Data Collection:** The input for the model can also be based on predictions and forecasts.

5. **Model Execution:** The provided input is used to execute the model with an appropriate algorithm.

6. **Result Evaluation:** Finally, the solution is classified as useful, improvable or useless.

2.8.1 The Transshipment Problem

The Transshipment Problem is a combinatorial decision problem that can be solved by means of OR methods. It forms the basis for the scheduling and routing decision problem proposed by this thesis, which will later be described in Chapter 6.

For a given instance of the Transshipment Problem, the goal is to find the optimal transportation routes between sources (the supply nodes) and sinks (the demand nodes) by using intermediate hubs. A problem instance is defined by a set of hubs connected by weighted edges that represent the distance between the two adjacent nodes. Depending on the definition, the set of sources and sinks can be disjunct. The goal for a given transportation network, supply and demand is to find the route(s) that minimize(s) the overall distance driven.

The Transshipment Problem can be formalized to a non-linear mathematical optimization model. Table 2.1 briefly describes all variables used in the model. (2.1) denotes the objective function to be minimized. It is the sum of distances of all driven tours times the cost. In (2.2) incoming and outgoing shipments are constrained. All incoming shipments subtracted by all outgoing shipments have to be equal to the balance. The balance specifies the supply and demand of each hub. (2.3) limits the capacity of hubs and (2.4) determines the number of trucks that is required to transport the given volume $s_{i,j}$.

Table 2.1: Variable Description

Variable Name	Description
H	Set of hubs
d	Duration matrix ($d_{i,j}$ = travel duration from i to $j : i, j \in H$)
t	Trucks on route matrix ($t_{i,j}$ = number of trucks from i to $j : i, j \in H$)
s	Shipment matrix ($s_{i,j}$ = shipment volume from i to $j : i, j \in H$)
$cost$	Cost per distance driven
$balance_h$, $h \in H$	Balance of hub h, > 0 for supply, < 0 for demand
$capacity_h$, $h \in H$	Hub capacity of hub h
$capacity_t$	Truck capacity

$$min_{x \in \mathbb{Z}^+} \sum_{i \in H} \sum_{j \in H} d_{i,j} \cdot t_{i,j} \cdot cost \qquad (2.1)$$

$$\forall i \in H : \sum_{j \in H} s_{i,j} - \sum_{k \in H} s_{k,i} = balance_i \qquad (2.2)$$

$$\forall i \in H : \sum_{j \in H} s_{j,i} \leq capacity_i \qquad (2.3)$$

$$\forall i, j \in H : t_{i,j} = \frac{s_{i,j}}{capacity_t} \qquad (2.4)$$

CHAPTER 3

Related Work

This chapter will provide an overview of recent contributions in the fields of research that are relevant to the thesis. This includes the Internet of Things in logistics (Section 3.1), event driven architectures (Section 3.2), autonomous control and decision support systems in logistics (Section 3.3) and algorithms for dynamic transshipment (Section 3.4). The review will be followed up by a short introduction of *ProveIT*[1], a project that is currently conducted at Forschungszentrum Informatik (FZI) in Karlsruhe.

3.1 The Internet of Things for Real-time Data Capturing in Logistics

Any type of decision support system for anticipatory logistics relies on a satisfactory level of real-time visibility of relevant resources [MGYA14]. Traditionally, this was achieved with passive identification technologies in the form of RFID. A number of surveys [MD16, DXHL14, SADP10, AIM10] have shown that the use of RFID in supply chain management has grown steadily. Applications for RFID include distribution and fleet management [WB12, NLW+12], as well as in-transit product visibility [DHS07, VLK07]. More recent work has investigated ways on how to extend the capabilities of RFID and EPC to overcome identification-only and inter-checkpoint limitations and to enable real-time tracking of resources. One major contribution, authored by Musa et al. [MGYA14], proposes "Smart tags" as an extension of RFID. These tags are small embedded devices that are attached directly to the cargo and use IP-based wireless communication to send data to service endpoints. The provided data depends on the configuration of the device, as the system architecture is modular. In addition to RFID for identification, the device can be equipped with location sensors (GPS, terrestrial), ambient sensors (e.g. humidity, temperature, etc.) and wireless network interfaces (e.g. 3G cellular, IEEE 802.11 Wi-Fi,

[1] http://prove-it.org

IEEE 802.15.4 WPAN, ZigBee, etc.). An embedded computational unit is used for information handling and the transmission of sensor measurements, but could also serve more sophisticated purposes. To that end, Musa et al. provide a conceptual framework for product intelligence of IoT technologies in logistics. This framework distinguishes four levels of intelligence, depending on the capabilities a technology provides. Each level assumes to also have the capabilities of all previous levels. The four levels are:

1. **Identification**: The product has a unique local or global identity (e.g. barcode or RFID).
2. **Data acquisition and handling**: The product can monitor its environment with sensors and communicate the measured data wirelessly.
3. **Problem recognition and reporting**: The product can identify problems such as system malfunctions or exceptions in the physical environment (e.g. temperature, location).
4. **Decision making**: The product is able to determine what needs to be done to remedy or handle exceptions and can give advice on its use, maintenance, recycling or disposal.

According to this classification, the most commonly used technologies today, RFID and EPC, would classify as first level product intelligence. Higher level technologies are not yet actively used in regular logistics, but are restrained to high-value products in the aviation and military industries. This is due to a number of limitations, mostly tied to costs and uninterrupted wireless communication [MGYA14].

The proposed framework in its initial configuration will work with devices with a product intelligence equivalent to RFID, i.e. will only require status updates for cargo at certain checkpoints (e.g. hub entry and hub exit) for routing decisions. However, by leveraging truck telematics and GPS location, rich data during the transit is available, allowing for routing decisions and rerouting of vehicles to happen while the vehicle is on the road. For usual distances between hubs, it might not be reasonable. To reroute vehicles, it is however a desired property for clusters of high activity and short hub distances (travel times < 1h), which also exist in the given transportation network.

3.2 Event-driven Architectures and Event Processing

Event-centric architectures for IoT based systems have been studied in recent literature. Liao et al. [LSSW11] present a real-time event streaming framework for RFID systems in retail. The framework supports real-time event queries as well as historic event queries from persisted events and serves as a generic source for real-time data, usable for both analytical and operational (e.g. optimization) purposes. The authors outline a benchmark by stating that even mid-size RFID deployments in retail would generate

vast amounts of data, at a rate of 1 TB per day, much of it being redundant. To avoid data redundancy, the authors propose a filtering mechanism for RFID event processing. Similarly, Rinne et al. [RSN16] propose a RFID-based logistics monitoring system based on CEP. The authors use patterns in events and time-window queries for counterfeit and theft detection with general cargo (i.e. small cargo loaded onto standardized containment units, e.g. palletes). Their work also includes a qualitative performance comparison of Esper[2] and Instans[3], two competing event processing platforms. The authors come to the conclusion that while both platforms are suitable tools for their intentions, Esper demonstrated clearly better performance and platform maturity. For this reason the framework presented in the thesis also uses Esper as a CEP platform.

It is worth noting, that both of the projects presented above consider only RFID-based events in their architecture, hence do not offer heterogenous event processing from different logistical objects and data sources. The framework presented in this thesis additionally considers telematic and location data from trucks as well as updates from traffic and weather services.

3.3 Autonomous Control and Decision Support

Autonomous control is an emerging field of research that addresses the increasing complexity of logistical processes [WH07]. In general, autonomous control attempts to manage logistical processes on an operational level, in a way that optimizes for certain criteria (e.g. costs, utilization) and can handle unforeseen events [SHG+10]. From an architectural point of view, this can be done either in a centralized or in a distributed way, where embedded devices, mounted on the cargo or a vehicle, are granted some autonomy in decision making and as a consequence, classify as fourth level product intelligence (as it was described in the previous section) [MGYA14]. DSSs are a predecessor of autonomous control. They also use methods of OR for decision making, but require human operators to make the final decisions and determine details, that have not been considered by the system [HRR00].

A number of projects for autonomous control and decision support in logistics have been published. Hu and Sheng [HS14] propose a DSS that aims at minimizing Empty Load Ratios (ELRs) in direct transport logistics using a multi-objective, mixed-integer linear optimization model. It monitors the state of vehicles and incoming transport orders and optimizes truck schedules by matching available trucks to compatible goods from orders. Aside from the primary objective function, which is to minimize for ELR, two additional objectives improve the matching rate and favor short travel distances. The authors present a sophisticated matching algorithm for cargo, that uses product compatibility matrices, integer range (e.g. temperature), and time window comparisons for compatibility checking. Similarly to the approach presented in this thesis, dedicated pickup and delivery times have to be considered by the optimization model. However,

[2] http://www.espertech.com/esper
[3] http://instans.org

the matching of general bulk cargo, as it is considered in the thesis, does not require the same degree of sophistication in product matching. The shipment size for this type of transportation typically does not exceed the standard dimensions of a EUR pallet.

Ngai et. al [NLW+12] designed a context-aware DSS for fleet management, that helps schedulers to make decisions in case of unforeseen events that cause delays. Such incidents include technical breakdowns, unplanned trips, delays and cancellations. The system generates an initial schedule for each day at a fixed time interval and reschedules in case of an unforeseen event. The incident handling is done by means of a predefined process that considers alternative actions depending on the context, modeled as a decision tree. Context information includes the position of vehicles, telematic data about the condition of the vehicle and eSeal[4] data about the status of transported goods. As an example, the countermeasures for a broken down vehicle would be different than for a vehicle that is stuck in traffic. Similar to this thesis, the project uses a cost-based objective function for scheduling. By using expenditures and revenue rather than utilization or ELR as the objective for optimization, a large number of factors in logistics processes (e.g. terminal handling, warehousing) can be quantized by means of a single metric and considered for optimization. With cost-based optimization, the utilization of resources is improved as a side effect, as good utilization generally is the cheapest way of operating those resources. Unlike the solution presented in the thesis, the system relies on a user interface and manual input for reporting incidents, rather than using CEP and IoT events.

Jedermann et al. [JCL+08] developed a system for distributed autonomous decision making in perishable goods transport. Rather than routing trucks centrally, the authors use embedded processing units deployed on trucks or food containers that act as autonomous software agents. This way, trucks themselves are able to change routing decisions. If the conditions change to be out of optimal, the agent can react autonomously to changes in the environment and the goods' shelf life. These agents use temperature and humidity sensors to monitor the state of the goods and track the shelf lives of the contained goods. The routing problem assumes trucks to be filled with goods that need to be delivered to a number of stores, where the loss (i.e. the goods with zero shelf-live) should be minimized. The distributed approach is based on the Traveling Salesman Problem (TSP) problem and uses two steps to find a solution. In the first step, a central routing agent computes an overall solution, where each vehicle is left with a number of options instead of one concrete route. Based on the options, the software agents can then decide to adapt their routes depending on the changing states of the goods.

3.4 The Transshipment Problem

Exact and heuristic algorithms of several variants of the Transshipment Problem (e.g. balanced, unbalanced, dynamic, time minimizing, fixed transshipment cost, restricted flow) have been proposed [EH08, LAP09, NT14, Khu15, HT01], with efficient polynomial time heuristics existing for some constrained problem classes [HT00].

[4] http://www.esealinc.com/

Herer and Tzur [HT01] propose the dynamic Transshipment Problem with deterministic demand over a finite planning horizon. This problem extends the original Transshipment Problem by adding a time dimension T with fixed periods $t = 1, ..., t_{max}$ and a dynamic, deterministic demand d_{it} at location i in period t. The authors represent the Transshipment Problem as a network flow problem and use a shortest path algorithm to find the optimal solution.

For a transshipment network with stochastic demand (i.e. predicted, not dynamic), Noham and Tzur [NT14] add considerations for fixed transshipment costs and provide a branch-and-bound heuristic to find an optimal routing policy, minimizing transshipment costs. Belgasami et al. [BSG08] use a genetic algorithm as a meta-heuristic for the transshipment problem with limited storage capacity and provide a specific implementation of recombination within the genetic algorithm.

Rais et al. [RAC14] consider a mixed integer model for solving the vehicle routing problem with transshipment. The model considers a heterogeneous, dynamically sized fleet and time windows for pickup and delivery. It provides a detailed description of the mathematical model and serves as a foundation for the model presented in this thesis.

While not all authors have published detailed evaluation data about their proposed algorithms, some benchmarks could be extracted. All of the presented algorithms were evaluated based on a network of four to fourteen nodes and a predefined cost model. The results of the branch-and-bound algorithm by Noham and Tzur showed a maximum gap of 0.08% to the optimal solution (in a network of ten nodes). Rais et al. achieved results with a maximum gap of 3.33%. The smallest CPU time for a network of fourteen nodes was 17 seconds, the highest was 18182.82 seconds. Unfortunately, no further remarks on the execution environment were found.

The optimization model proposed in this thesis shares certain characteristics with the model proposed by Rais et al., but will additionally consider deadline penalties, loading gate restrictions, as well as the results of previous optimization cycles for recalculating the schedule. Typically, models solving the Transshipment Problem return the number of vehicles that is required to distribute the cargo in order to correctly match supply and demand, similar to the introductory example provided in Section 2.8. The proposed solution formulates a second problem, that is used for assigning vehicles to the previously computed vehicle demand. This aspect is a major requirement for the practical application of vehicle routing in logistics, as some additional constraints for vehicles might apply. As an example, the organizational form of the tested transportation network requires vehicles to revisit their home hubs after certain periods of time, as drivers need to return home after their shifts and trucks from external trucking companies are not provisioned permanently. These constraints are modeled as a second optimization problem, using the output of the first model and assigning vehicles to the vehicle demand. Details about the two proposed models will be covered in Chapter 6.

3.5 FZI ProveIT

ProveIT („Production plan based recovery of vehicle routing plans within integrated transport networks“) is a project in development at FZI (Forschungszentrum Informatik) in Karlsruhe that attempts to achieve similar goals to the solutions presented in this work. The project is backed by industry partners such as Bosch, ZF, and Geis Global Logistics, as well as the German Federal Ministry for Economic Affairs and Energy[5]. It is one of the most recent projects in operational management of transportation networks. *ProveIT* uses data from a mobile application, operated by the vehicle drivers and integrated information from business partners for vehicle routing and incident handling. In contrast to the projects presented in the previous sections, *ProveIt* aims at addressing all aspects of a logistics process, providing high levels of autonomy from order placement to delivery. A pilot of the project is currently evaluated by the sponsoring companies. The efficiency gain, regarding energy consumption, costs, and emissions, is expected to be around 5 percent.

[5]Unfortunately, no publications for the project yet exist. In addition to the information available at the website `http://prove-it.org/`, the information presented was gathered in an interview with Dr.-Ing. Iris Heckmann, who leads the Department for Information Process Engineering at FZI. The interview was conducted on September 20th, 2016 in Karlsruhe

CHAPTER 4

Methodology

In this chapter, the chosen methods and techniques for building, designing and evaluating the proposed framework and optimization model will be discussed. This includes architectural guidelines for event-driven architectures, DSSs and software agents, as well as some key properties of the considered transportation network.

4.1 Properties of the Considered Transportation Network

The available transportation network data represents small to mid-size transportation networks, as they are operated by many European logistics providers [Gud12]. With a size of eleven hubs it is consistent with transportation networks considered in related work, such as by Noham and Tzur [NT14] or Rais et al. [RAC14]. The transportation network is based on the general cargo (LTL) road network of DB Schenker AG in Austria, comprising eleven branches in all federal states of Austria and shipping a total volume of 9 Mio. tons over their road transportation network per year [Sch16]. About 370000 tons (4% of the total volume) are shipped via LTL and general cargo logistics services, which is the focus of this thesis. The volume breaks down to approximately 2 million transport orders per year (about 30000 - 40000 orders per week)[1]. Typically, these orders are processed within 24-48 hours.

The map depicted in Figure 4.1 provides an overview of all Austrian branches and the main hub connections (non-exhaustive). The thickness of connections indicate the transported volume transported over that connection. According to the analyzed test data set, one particular region of very high activity was identified. This cluster of hubs is in the mid-western part of Austria and comprises four hubs that all have a travel time < 1h to any adjacent hub within the cluster and particularly high order volumes.

[1]The numbers were derived from the test data set provided by DB Schenker AG.

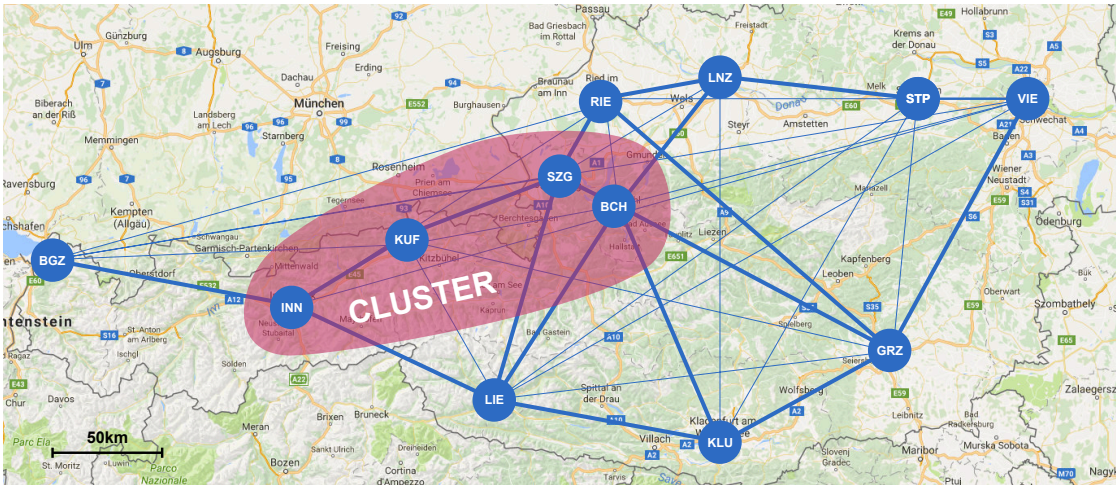


Figure 4.1: Geographical overview of the considered transportation network

DB Schenker AG does not operate its own vehicle fleet, but uses third-party freight companies for the traffic between branches. It is important to note that despite using different companies, the vehicle fleet is homogeneous to a large extent, which is an important factor for optimization. The number of trucks in operation for the network depends on the time of the year, but can be estimated to be around 200 - 300[2]. More details about the test data set will be presented in Chapter 7.

4.2 Event Load Estimation

As described above, the considered transportation network consists of eleven hubs ($|H| = 11$), a fleet of about 300 trucks ($|Tr| = 300$) and a shipment volume of about 40000 shipments per week ($|S_w| = 40.000$). If we assume a truck location update frequency $f_{tr} = \frac{1}{10s}$, a hub status event frequency $f_h = \frac{1}{30s}$, an average turnover rate $p^s_{avg} = 2.5$ per shipment and two scan events per shipment and turnover, an average number of line hauls per day for a truck $p^{tr}_{avg} = 5$ and two docking events per line haul, the event frequency f can be calculated by 4.1.

$$f = |Tr| \cdot f_{tr} + |H| \cdot f_h + |S_w| \cdot p^s_{avg} \cdot 2 + p^{tr}_{avg} \cdot 2 \tag{4.1}$$

For the given assumptions, this results in $f = \frac{30}{1s}$ or about 106000 events per hour that need to be processed by the framework.

[2]The information was acquired in a series of interviews during my work at DB Schenker AG.

4.3 Architectural Guidelines

Certain principles and best practices have been incorporated in the design of the system architecture. These principles have been proposed for similar settings, where a system needs to observe certain aspects of a dynamic environment (e.g. a transportation network) and manipulates it by applying its decisions back to the environment [PGM12, LSSW11, NLW+12]. The guidelines will be presented in two parts: The first part will discuss desirable characteristics for real-time DSS, the second part will cover the architecture of software agents.

4.3.1 Guidelines for Real-time Decision Support Systems

Pillac et al. [PGM12] propose three desirable (fairly generic) characteristics for real-time DSS in dynamic vehicle routing. These characteristics are:

- **Event-driven:** The architecture should promote events (i.e. messages) as the main means of communication to model state changes in the data. Instead of receiving updates periodically, the available data for decision making is updated for every event, as real-time DSS should offer a consistent, up-to-date representation of the environment.
- **Parallelized:** The underlying architecture should be parallelized and should take advantage of distributed computing architectures in order to perform several tasks in parallel. This allows for a large number of events to be processed and for decisions to be made fast.
- **Flexible:** The logistics landscape is vast and changes quickly. Thus, DSS should be based on universally valid assumptions and be extensible for specific aspects of different applications and an evolving field of applications and technologies.

The proposed framework aims to achieve all three of the named characteristics. It is inherently based on events as they are the major means of communications for the system with the environment, as well as for software components within the system. Secondly, it allows for a high level of parallelism, by using a multi-threaded environment for event processing and a separate process for the software agent, which is responsible for mathematical optimization. Third, all assumptions regarding the architecture and the transportation network (i.e. considered events) have been chosen to represent the problem domain as generically as possible. The application is highly parameterized and can be easily tailored to specific transportation networks and IoT devices.

In addition to the three characteristics presented above, Ngai et al. [NLW+12] framed a set of functional and non-functional requirements for real-time DSS. In the following, the list of requirements will be presented, each supplemented with an explanation, of how these goals are met by the design of the proposed framework.

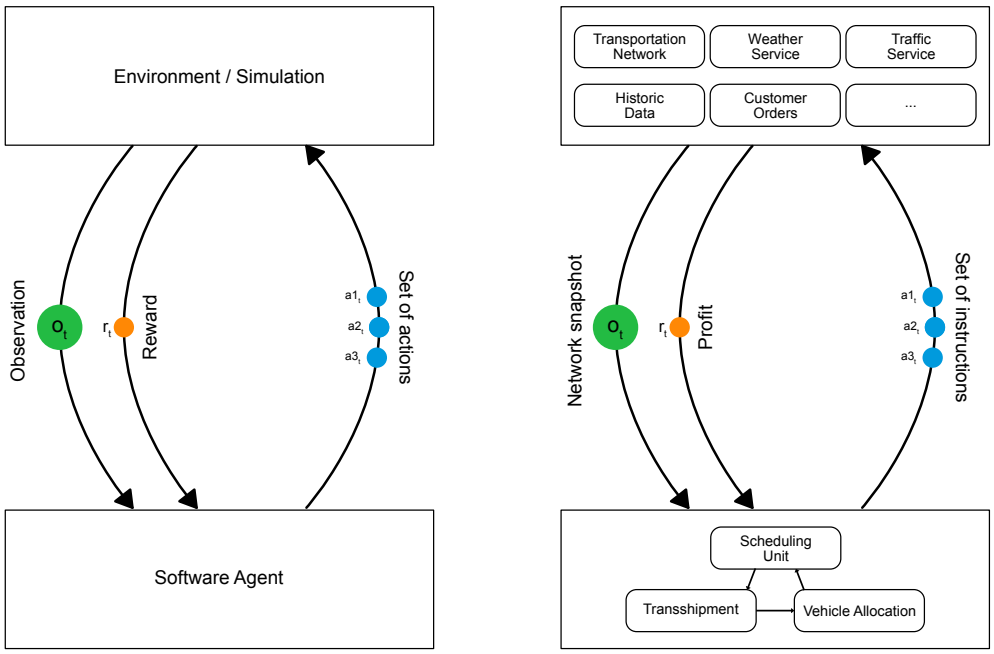
- **Operate in real-time and provide prompt responses:** While regular planning cycles may be done overnight and last several hours (e.g. to compute the preliminary schedule for the next day), unexpected events need to be dealt with promptly. In this case, rescheduling cannot wait for a full evaluation, which likely takes hours to compute with mathematical optimization, if it is done from scratch. Thus, rescheduling has to be modeled in a way that can leverage and alter existing solutions, which is sometimes referred to as a warm start [DDKS15]. The proposed optimization model can use an existing schedule as input for a new calculation. This tremendously speeds up the calculation of an adapted schedule.

- **Be able to handle different kinds of unexpected events:** In this work, only a small number of unexpected events are covered, all of which affect the travel duration between hubs. Three main cases can be distinguished: Cases where only single trucks are affected (e.g. technical breakdown), cases where all trucks of a connection before a certain point on the road are affected (e.g. traffic jam), and cases where all trucks on a connection are affected (e.g. bad weather conditions).

- **Consider different contexts when deciding on how to react to unexpected events**: The subsystem for unexpected event handling considers different contexts and determines appropriate actions by means of a predefined process. The context is mostly defined by the current schedule, but also takes sensory data into account. A truck delay might not have any impact on the schedule, if all shipments reach their connection in time and the truck can execute its next instructed movement without delay. Other cases might allow some instructions to be postponed without any consequence to the rest of the schedule. Finally, if all other options are exhausted, a rescheduling cycle (in the warm starting mode mentioned above) is triggered. Details about the subsystem for unexpected event handling will be discussed in the next chapter, in Section 5.8.

- **Distribute responses to all necessary parties accurately, effectively, and quickly**: The technology stack that was chosen to implement and deploy the framework (Java 8, Spring Boot[3], IBM Bluemix[4]) provides a vast number of options for IP-based communication. The framework uses web sockets to promptly display changes in the web user interface and exposes a RESTful API that opens up all major functionalities of the framework for potential partner applications in a very interoperable way.

[3]https://projects.spring.io/spring-boot/
[4]https://www.ibm.com/cloud-computing/bluemix/



(a) Generic case

(b) Application specific case

Figure 4.2: Feedback loop of software agents interacting with the observed environment.

4.4 Software Agents

Systems, where a (semi-)autonomous entity observes and changes the environment based on a set of rules are referred to as agent-based systems [VdHW08]. Software agents are used in many disciplines, including chat applications and personal assistance (e.g. chat bots), game theory, operations research and swarm intelligence [Nwa96]. By designing the architecture of the framework around the concept of a software agent, the concerns of monitoring the network and optimizing the schedules (i.e. the two parts of the proposed solution) are cleanly separated. In this particular case, a software agent is used to decide which actions to take in order to complete the transport orders in the most cost-efficient way, based on the current state of the transportation network. Figure 4.2 illustrates this basic feedback loop, both in the general case on the left (Figure 4.2a) and in the domain specific case on the right (Figure 4.2b). The framework's role is to provide the agent with accurate observations of the environment, both in terms of data richness and freshness.

4.4.1 Definition

Formally, a software agent interacts with its environment in discrete time steps. At each time step t, the agent receives an observation o_t which is a subset of the current state $s_t \in S$ of the environment, where S is the set of all possible states. Based on the observation, the agent selects an action $a_t \in A(s_t)$, where A denotes the set of all available actions in the state s_t. By setting an action, the environment transitions into a new state s_{t+1} and the agent is signaled a certain reward by the environment. As an example, the environment could be a transportation network, an observation could be all sensor data collected with IoT devices and the reward could be the profit made by successfully fulfilling transport orders. Possible actions would then include the loading and unloading of cargo to trucks and the movement of trucks between hubs. In general, the goal of an agent is to maximize the received reward over time [SB98]. This is done by following a certain policy, i.e. a set of rules or mechanisms $\Pi_t(s, a)$, that maps states to actions which are likely to maximize the reward. Strategies that focus on short term rewards might not help the overall goal. The challenging part of defining a policy is to find actions that maximize the long term reward, even though they might not be immediately beneficial in the short-term.

Agents can fully ($o_t = s_t$) or partially ($o_t \subset s_t$) observe the environment, depending on their complexity and nature. In the given case of a transportation network the observation is only partial, since never all possible nuances of reality can be sensed by electronic sensors. Since unexpected events cannot be prevented by the agent, actions not always deterministically change the state of the environment [SB98]. In the case of a transportation network, the probability of unexpected events is rather high compared to more deterministic settings, where the environment is restricted and actions happen deterministically. For example, a movement in chess is certainly followed by a state, where the chess piece is in its new position. However, a movement action for a truck in a logistics network does not imply that the truck will depart properly, much less arrive at the destination on time. Those state transitions can only be assured to a certain degree.

In the following two chapters, the two main contributions of the thesis will be presented. The next chapter will introduce a framework for optimizing freight streams based on transportation network snapshots and IoT data. Chapter 6 will provide an implementation of a routing policy $\Pi_t(s, a)$ for the software agent, based on mathematical optimization of the Dynamic Transshipment Problem with time constraints.

CHAPTER 5

A Framework for Real-time Optimization of Freight Streams

This chapter is dedicated to the first of two contributions of this thesis, a framework for optimizing freight streams in transportation networks. The primary function of the framework is to consolidate data from heterogeneous data sources in order to maintain rich information about the state of the transportation network. It is characterized by three fundamental concepts: An event-driven architecture, a software agent for optimizing freight streams and a subsystem for unexpected event handling. These concepts, together with a snapshot-based data access model and a service for future network state prediction will be elaborated.

5.1 Architectural Overview

Figure 5.1 provides a top-level view of the proposed architecture. Its main components are: The core framework (1), the software agent (2), and the environment (3), which, for the purpose of evaluation, is replaced by an environment simulation application.

(1) can further be broken down into the modules for event processing (Section 5.4), a prediction service for future states of the transportation network (Section 5.7), a subsystem for unexpected event handling (Section 5.8), and an agent intermediator (Section 5.5), responsible for communicating and triggering the software agent.

(2) is responsible for dynamically scheduling vehicles and routing shipments. Its consists of a transformation unit to convert observations into mathematical models in a pre-processing step (i.e. transforming Java objects to OPL[1], the required input format for

[1] https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.studio.help/pdf/opl_langref.pdf

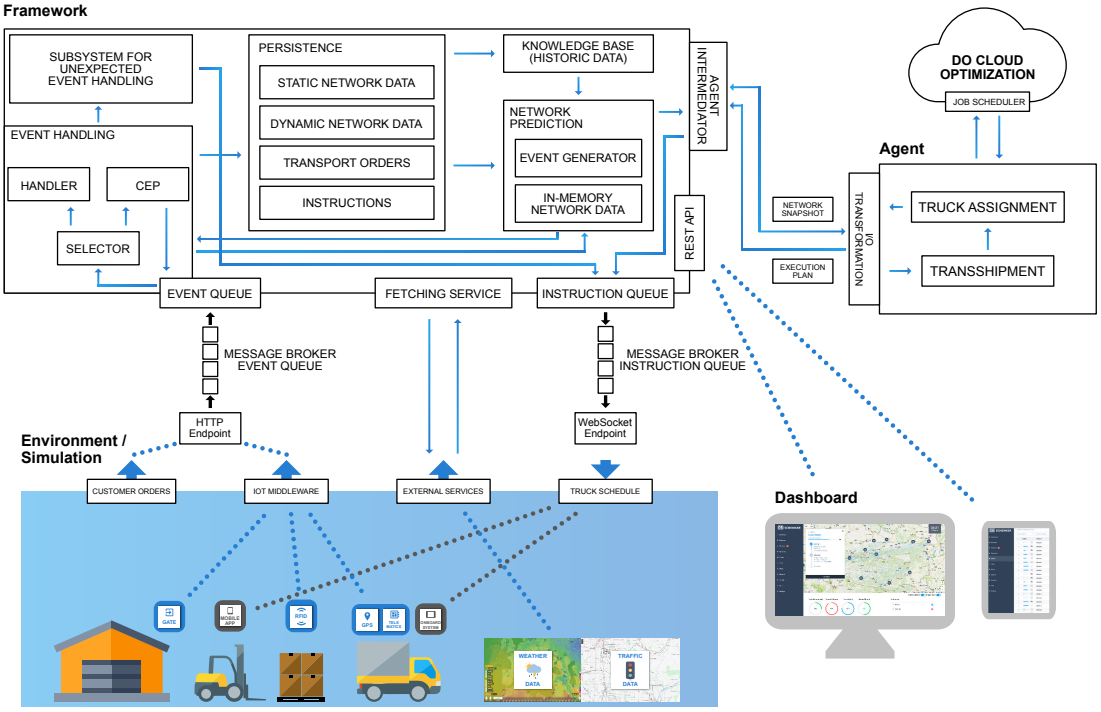


Figure 5.1: Overview of the proposed framework architecture

CPLEX) and solutions into instructions in post-processing. The models are solved by *IBM Decision Optimization Cloud*, a cloud service for solving *CPLEX* models. Details about the agent will be discussed in Section 5.5.

(3) can be defined as every entity, system or service that is capable of sending events to the event queue. This includes all sensory data from IoT devices deployed to cargo, warehouses and trucks, new transport orders from customers and external services (e.g. weather and traffic services). For the purpose of evaluation, the environment is simulated by a separate application. This simulator can mimic incoming transport orders, moving vehicles and the loading and unloading of cargo. In general, the simulator generates events deterministically, based on the instructions provided by the software agent. It is however also possible, to simulate unexpected events such as weather warnings, traffic congestion and technical breakdowns of vehicles. The detailed event model will be presented in Section 5.3. Details about the simulations will be discussed in Chapter 7.

5.2 Data Model

The class diagram depicted in Figure 5.2 provides an overview of the main entities and attributes defined in the data model. The main entities of a transportation network are `Hub`, `Truck`, `TransportOrder`, `Connection` and `Cargo`. Hubs are constrained by their

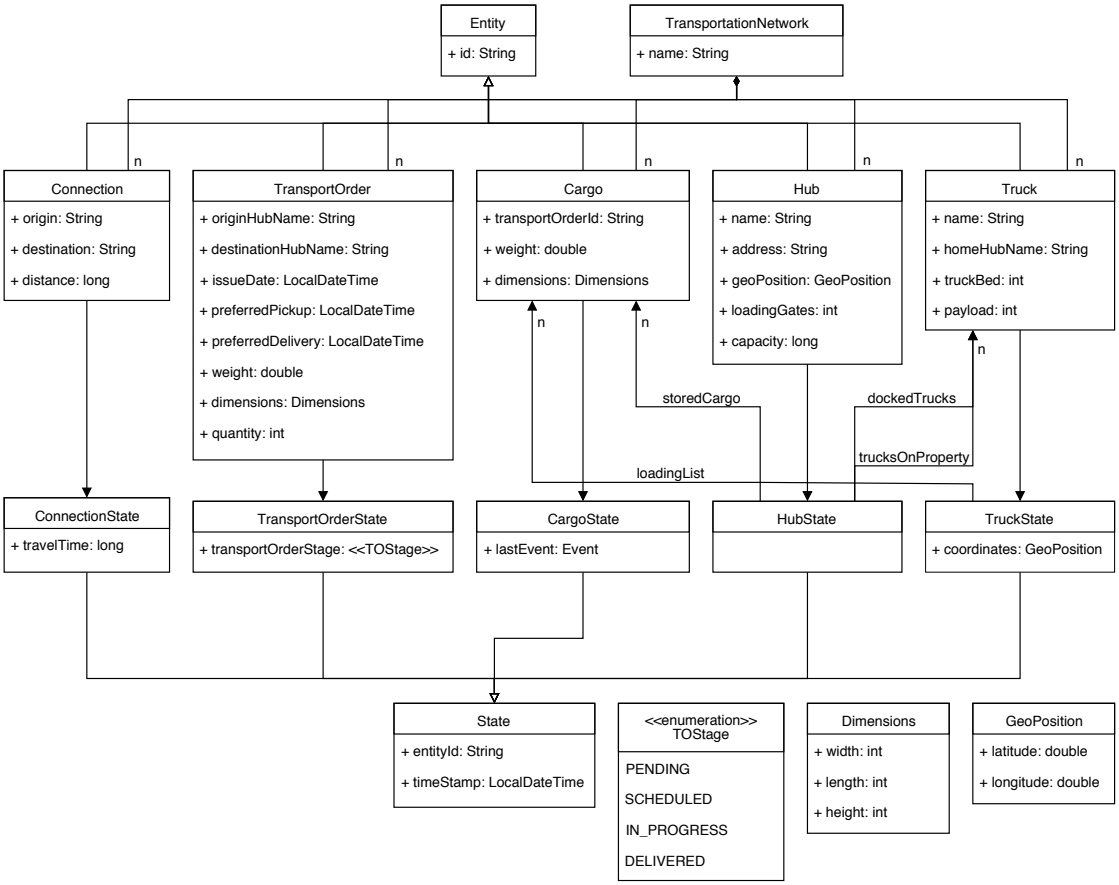


Figure 5.2: Class diagram of the transportation network entities

overall capacity and their state holds two separate lists of trucks: `trucksOnProperty` lists all trucks that are within immediate proximity of the hub (e.g. are parking on the property) and `dockedTrucks` lists all trucks that are currently docked to a loading gate. Additionally, the `HubState` holds the list of the currently stored cargo. The throughput is limited by the number of loading gates. In a similar fashion, trucks are limited by their overall payload, measured in loading meters, since the shipments are transported via standardized EUR pallets. A truck state holds the current list of cargo that is on the truck as well as its current GPS location. Transport orders specify the cargo dimensions, weight, and quantity, as well as an issue date and dates and locations for pickup and delivery. The respective state holds the processing stage of the order. Cargo entities store its dimensions and weight and reference the transport order. Cargo states store cargo related events that happen during the shipping process. By keeping a history of cargo events, all steps of the shipping process can be retraced for each transported cargo (e.g. `[<2016-1-1:07:20:38,EXIT,HUB-A>,<2016-1-1:09:58:05,ENTRY,HUB-B>]`).

Separation of static and dynamic data

As it can be seen in the class diagram, the model distinguishes static (or immutable) data stored in the entity classes and mutable (or dynamic) data stored in the state classes and referenced by the entity. The distinction between static and dynamic attributes is done in the following way: Each attribute, that can be altered by events is considered dynamic and is stored as an entity state. This approach allows for the storage of multiple states for each entity in order to track their history. In addition to the entities above, the data model includes events (Section 5.3) and instructions (Section 5.6).

5.2.1 Data Access Model: Snapshots

In order to get a holistic, up-to-date view of the transportation network (i.e. have access to all data relevant for optimization), the persistence layer is able to fetch all entities of a transportation network at once, providing a single object containing all entities of the network at their most recent states. As these information objects always hold the state of a particular time instant, they are referred to as snapshots and are represented as `TransportationNetwork` objects. Holding these objects in memory is feasible, because completed transport orders and their associated cargo objects are not contained in a snapshot. No events, except for the events referenced in the cargo state, and no instructions are stored in a snapshot. Figure 5.3 provides a memory profile, showing that the relevant data for optimization, i.e. the data stored in a snapshot, is only a small fraction of all collected data over time. For the simulations that were performed, simulating one week of network activity with about 35.000 transport orders, the size of snapshots did not exceed 20 MB. As an alternative, it is possible to access the database directly by the agent by granting him reading permissions. Reasons for why the snapshot approach was chosen in favor to direct database access will be provided in Section 5.7.

Filtering past orders and instructions and only keeping the most recent state of objects does not weaken the information available for the decision making process. Omitting these large amounts of data is possible, because a transportation network snapshot has the Markov Property in respect to the presented software agent.

The Markov Property

According to the definition of Sutton and Barto [SB98], an observation is said to have the Markov Property, if it succeeds in retaining all relevant information, i.e. retaining information also from previous observations, that are relevant in the decision making process. In a chess game, for example, the current configuration of all pieces on the board fulfills the Markov Property, even though the sequence of moves is not kept explicitly in this representation. In the specific case of the thesis, where an observation corresponds to a transportation network snapshot, previous movements and transport orders do not influence the planning of future movements. All that is required is the current state of all involved entities (hubs, trucks, connections) and currently open transport orders.

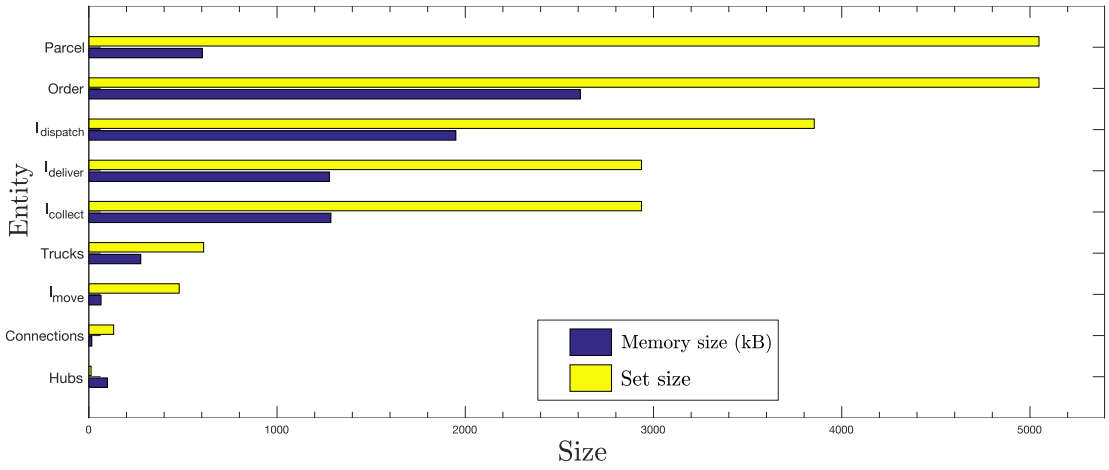


Figure 5.3: Memory profile of a transportation network snapshot

Assuming that an observation does not fulfill the Markov property, then the expected reward r for a taken action A_t at the current State S_t depends on the entire sequence of actions and states, formally denoted as:

$$Pr\{R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \tag{5.1}$$

If, on the other hand, the observation has the Markov property, the environment's response at $t + 1$ only depends on the state and action representations at t, These dynamics can be defined by specifying only the most recent state and action pair:

$$Pr\{R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t\} \tag{5.2}$$

The Markov property is fulfilled by snapshots in respect to the given software agent, because the agent does not require historic states for maximizing the expected reward. This property allows for omitting large amounts of data and make snapshots a justifiable representation of observations for the software agent.

5.3 Event Model

Figure 5.4 shows the class diagram for all events, grouped by originating entity. The model distinguishes primitive and complex events. Primitive events are emitted by an entity of the environment, complex events – marked with an asterisk – result from CEP queries. The framework uses a number of CEP queries to derive higher-level knowledge from primitive events or aggregate certain values over time. For example, one query is used to monitor the incoming transport order volume: By default, the optimization

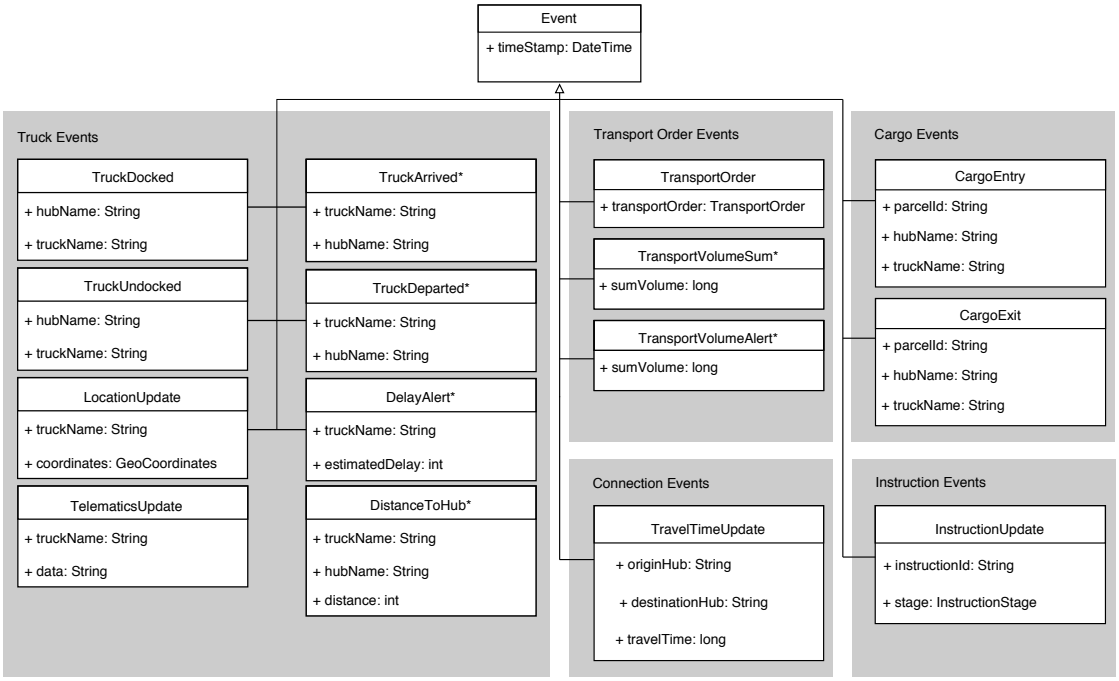


Figure 5.4: Class Diagram of primitive and complex events. ("*" denotes complex events)

agent is triggered in fixed periods (e.g. daily). If however the demand of transport volume exceeds a certain threshold for a certain time, i.e. many transport orders specify similar preferred pickup times, an extra optimization cycle is triggered by the agent intermediator. Table 5.1 provides a description of each event.

Updates from the traffic and weather service are not represented as dedicated events, but rather influence the state by changing the expected travel times of connections. As there is no standardized message protocol for data exchange, a small adapter service periodically fetches traffic and weather data updates and transforms them into appropriate `TravelTimeUpdate` events. The data sources that have been used for traffic and weather information will be described in Subsection 5.5.3.

5.4 Event Processing

All events originating from the environment, i.e. all primitive events, are initially queued in a message queue that is accessible via an IP endpoint. For this reason, IP-based communication is the only prerequisite for sensor devices to communicate with the framework. The framework uses RabbitMQ[2] as a message broker. Each event – primitive or complex – is processed by a dedicated handler and triggers the logic that is

[2] `https://www.rabbitmq.com/`

Table 5.1: Description of all events in the event model (primitive and complex)

Event	Description
`TransportOrder<o>`	A new transport order o was issued
`CargoEntry<c,h,t>`	Cargo c was scanned entering hub h loaded off truck t
`CargoExit<c,h,t>`	Cargo c was scanned leaving hub h loaded onto truck t
`TruckDocked<h,t>`	Truck t docked to a free gate of hub h
`TruckUndocked<h,t>`	Truck t undocked from a gate of hub h
`LocationUpdate<t,p>`	Truck t has a new GPS position p
`TelematicsUpdate<t,d>`	Truck t has new telematics data d
`TravelTimeUpdate<c,d>`	Connection c has a new estimated travel time of d minutes
`*DistanceToHub<h,t,d>`	The distance between truck t and hub h is d (derived from `LocationUpdate`)
`*TruckArrived<h,t>`	The distance between truck t and hub h fell below a certain value (derived from `DistanceToHub`)
`*TruckDeparted<h,t>`	The distance between truck t and hub h exceeded a certain value (derived from `DistanceToHub`)
`*DelayAlert<t,d>`	The scheduled arrival of truck t will be delayed for approximately d minutes
`*TransportVolumeSum<s,t>`	The sum of collected order volumes at time t is s (derived from `TransportOrder`)
`*TransportOrderVolumeAlert<s,t>`	The sum of collected order volumes after the last scheduling cycle exceeded a certain value (derived from `TransportOrder`)
`*InstructionUpdate<i,s>`	Instruction i changed to state $s \in \{$ `pending`, `in progress`, `completed`$\}$

associated with the event. Some selected events are additionally processed by the CEP platform Esper. Esper is specifically used to cover three use cases that require aggregated knowledge, derived from a stream of previous events. The first use case has been briefly discussed in the previous section: Esper is used to monitor incoming transport volume, in order to detect future utilization peeks as early as possible and adapt the schedule if necessary. The second two use cases demonstrate basic examples of geofencing with Esper. One example detects unusually long standing times based on GPS trails. These standing times might indicate breakdowns or delays of vehicles. The second example uses calculated proximity to hubs in order to determine their arrival status. If the vehicles are located within a certain radius to the hub, they are considered available at that hub. This allows the distinction between docked trucks and available trucks, as not all trucks might have a free terminal gate to dock to. Ideally, vehicle telematics and dedicated sensors at the property gate would generate explicit events for these occasions, but these sensors are often unavailable in current transportation networks. The three implemented CEP queries merely serve as examples for possible use cases of CEP in road transportation. The capabilities of the framework can easily be extended by providing additional queries and event handlers.

5.4.1 Monitoring Incoming Transport Order Volume

Continuously collecting data about incoming transport orders helps to quickly derive implications to the schedule. In general, the software agent optimizes the schedule in fixed intervals with a default value of 24 hours, i.e. the schedule is adapted once a day (this does not include the handling of incidents). If new transport orders with shorter lead times than 24 hours are issued, the schedule might be adapted, considering the new orders. If, for example, free transport volume is available on existing movements, these orders are dispatched onto existing movements, to better utilize already provisioned vehicles.

Listing 5.1 provides the Esper query that has been used to monitor the incoming order volume. It uses `TransportOrder` events as input and produces `TransportOrderSum` events as output. In the query, `ts` denotes a dedicated time service implemented in Java, where `ts.before_inv(x)` returns `true` for dates that happen before the next invocation of the software agent and `ts.time_h(x)` returns the time of `x` in the granularity of hours. The latter function is used to group the incoming volume by the hour of arrival, in order to detect times of high and low utilization. The resulting event stream triggers a second query that checks for the volume limit $limit_v$ to be exceeded. If the limit is exceeded at any given time, a `TransportOrderVolumeAlert` is issued.

Listing 5.1: Monitoring the weights of incoming orders within invocation intervals

```
expression filter { x => ts.before_inv(x)},
expression pickup { x => ts.time_h(ts.now,x)}
sum(transportOrder.weight) as sumWeight
from TransportOrder.win:time_batch(ts.now, ts.inv_interval)
where filter(preferredPickup)
group by originHub, pickup(preferredPickup)
```

5.4.2 Detecting Irregularities in Location Patterns

In order to detect unusually long standing times of vehicles, e.g. during unreported traffic congestion or vehicle breakdowns, a query for detecting irregularities the sequence of GPS location updates is provided. For the purpose of readability, the example depicted in Listing 5.2 was condensed to two consecutive location events. This would be impractical in a realistic scenario, as short stopping times, e.g. for traffic lights, should not produce a warning. Depending on the update frequency for locations, the number of events in the pattern has to be adapted. For the given assumptions of $f_{tr} = \frac{1}{10s}$, a low two-digit number might be appropriate. The query uses a computed distance between the consecutive locations of a truck to detect movement or standstill, provided by `Geo.dist(latX, longX, latY, longY)`.

Listing 5.2: Detection of GPS sequences for non-moving vehicles

```
select * from LocationUpdate
match_recognize (
partition by truck
measures
A.location.latitude as latA,
A.location.longitude as longA,
B.location.latitude as latB,
B.location.longitude as longB,
pattern (A B)
define
B as Geo.dist(latA,longA,latB,longB) < limit
```

5.4.3 Detecting Arrivals and Departures

Based on the `DistanceToHub` event, which is calculated for a subset of `LocationUpdate` events, Esper signals `Arrival` and `Departure` events for trucks depending on their proximity to hubs. The query depicted in Listing 5.3 signals an arrival, if the distance between a truck and a hub falls below a certain radius. To prevent consecutive arrival events for each location update within the radius, the second line of the query ensures,

that only the first event with a distance smaller than the radius triggers an arrival. The event query for departures is analogous.

Listing 5.3: Detection of arrivals based on the calculated distance to hubs

```
select * from pattern[every
a = DistanceToHubEvent(distance > radius) ->
b = DistanceToHubEvent(distance < radius, b.truck = a.truck, b.hub =
    a.hub)]
```

5.5 A Software Agent for Dynamic Vehicle Routing

The discussion about the software agent, which is the central unit of decision making, will be split into two parts. The first part, covering architectural aspects of the agent, will be discussed in this section. The second part will cover the underlaying routing policy in detail and will be discussed in Chapter 6.

5.5.1 Overview

All of the presented topics so far, including the data model and data access, as well as events and event processing, have the fundamental goal of providing the software agent (and the subsystem for unexpected event handling) with rich and up-to-date information. The agents responsibility is to then take the current state of the network and provide a vehicle schedule to process all pending transport orders. The subsystem for unexpected event handling and the software agent are the only units that can issue instructions and thereby actively influence the environment. The difference lies in the temporal scope of decision making. While the agent makes planning decisions, focusing on long-term efficiency, the decisions made by the subsystem only influence immediate outcomes, triggered by unexpected incidents.

5.5.2 Framework Communication

Listing 5.4: The Software Agent Interface

```
interface Agent {
   schedule(snapshot: TransportationNetwork) : List<Instruction>;
}
```

Whenever the agent intermediator triggers the agent, it transmits a transportation snapshot to the agent. The agents' job is to then provide a mapping from the snapshot to instructions, i.e. a routing policy $\Pi_t(s, a)$. The implementation of the policy is not limited to using mathematical optimization techniques, as long as it can adhere to the

basic communication interface defined in the feedback loop, i.e. snapshots as input and instructions as output. For the reason of adherence to this interface, the implementation of the agent is restricted to a single public method, defined in Listing 5.4. This method is used by the agent intermediator. The intermediator is responsible for triggering and communicating with the agent, as well as fetching new instructions and handle further distribution to appropriate channels of the environment.

5.5.3 Input Sources

As depicted in Figure 4.2, observations consists of several input sources. Those input sources can be classified into three different categories:

- **Sensor data (internal environment)**: The most important observational data is the current network state, which is kept in a centralized data store, updated by events from the sensor framework. The distinguishing factor of this category is the internality of the data. This means that the observed data is mostly the consequence of actions taken by the agent. This is in contrast to external events, which the agent cannot influence.

- **Information services (external environment)**: Besides the internal state of the network, services that predict or report external events - i.e. events that cannot be controlled by the agent - can be valuable sources of information. The proposed agent uses two external data services to predict travel times between hubs, namely the Google Distance Matrix API[3] for computing dynamic travel times between destinations and the Weather Company Data for IBM Bluemix API[4] for weather alerts. Traffic and weather are major factors regarding travel times and travel times are essential factors for vehicle routing, since the overall shipment of a cargo usually consists of several trips to intermediate hubs. Travel times (and delays of such) will determine, whether a shipment can reach its connection transport or not.

- **Knowledge base (context data)**: The knowledge base forms the third pillar of observed data. It consists of data that is helpful in determining future demand. Examples include historic data about the transported cargo volume (e.g. to determine the demand based on the demand of the last years around this time), patterns in transport orders of customers, data curated by experts or calendrical data (e.g. information about public holidays). While the framework considers a knowledge base as observational data to be integrated optionally, the implemented prototype and the mathematical optimization model do not include stochastic demand prediction, but solely work on the ground of existing transport orders and instructions.

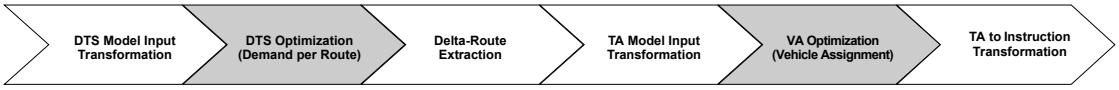


Figure 5.5: Overview of the optimization pipeline. The shaded steps are solving mathematical optimization problem.

5.5.4 Optimization Pipeline

On an architectural level, the implementation of the routing policy can be conceptualized as a pipeline. The agent needs to process all steps in the pipeline, given the input data (i.e. a transportation network snapshot) to return according instructions. Figure 5.5 illustrates this pipeline. The two main steps (shaded in the graphic) correspond to solving the according two optimization problems for obtaining a schedule, namely calculating a vehicle demand in the first step and assigning concrete vehicles to the demand in the second step. The steps in between serve as transformations between the different required input formats. In the first step of the pipeline, the given transportation network snapshot is converted from a JavaBean object into the Optimization Programming Language (OPL) data format, which together with the OPL model forms the optimization problem that is solved by the *CPLEX* model solver. The hosted *CPLEX* solver *IBM Decision Optimization on Cloud* was used for the implementation. The solution returned by the service is represented in the JSON data format and lists the number of required vehicles and shipped order ids for each connection and time interval. As the input for the model also includes all movements from the previous calculation cycles, for assigning vehicles to the demand, the difference from previous to newly required vehicles needs to be calculated. These delta values, as well as the state of all vehicles and their future movements, are then used as input for the second optimization problem, that assigns available vehicles to the vehicle demand. Once all demanded movements have been assigned a vehicle, the dispatch instructions for the cargo to the respective vehicles can be created.

Listing 5.5 provides an example for input orders passing the pipeline. In the example, `snapshot.orders` hold the two sample orders that need to be transported and are contained in the transportation network snapshot. The first order requires 25 volume units to be shipped from *VIE* to *INN*, starting at time $t + 0$ with a deadline at $t + 10$ hours. The second order requires an equivalent transport from *LNZ* to *INN*. The result of the first optimization results in the set of required movements for transportation, denoted as `demand`. In the case of the given example, two separate movements are required. One from *VIE* to *LNZ* and a separate one from *LNZ* to *INN*. Movements specify departure and arrival times (i.e. $t + 1$ and $t + 4$ respectively for the first movement), as well as all dispatched orders and the number of required vehicles. The demand is finally mapped to actual vehicles in `assignment`, by solving the second optimization problem for vehicle assignment to movements. The results corresponds to a preliminary schedule, that

[3] `https://developers.google.com/maps/documentation/distance-matrix/`
[4] `https://twcservice.mybluemix.net/rest-api/`

provides operational instructions for all movements within the transportation network to complete the orders.

Listing 5.5: The Software Agent Interface

```
> snapshot.orders
[
   {"order-a", 0, VIE, INN, 25, 10},
   {"order-b", 0, LNZ, INN, 25, 10}
]
> demand
[
   {"VIE", "LNZ", 1, 4, {"order-a"}, 1},
   {"LNZ", "INN", 6, 9, {"order-a", "order-b"}, 1}
]
> assignment
[
   {"V-VIE_1", "VIE", "LNZ", 1, 4, {"order-a"}},
   {"V-LNZ_1", "LNZ", "INN", 6, 9, {"order-a", "order-b"}}
]
```

5.6 Instruction Model

Instructions (i.e. actions in the software agent nomenclature) are a one-way communication channel from the agent to the environment. When the agent issues an instruction, it expects the environment to react in a certain way. For example, if the agent issues a movement instruction for a truck, it expects an actual movement from that truck which it could then observe via certain events, e.g. an undocking event from the origin hub, followed by several location events and a final docking event at the destination hub. For the given transportation network scenario, six different types of instructions have been created, each of which yield a certain expected reaction from the environment.

- **Collection instruction**: A collection instruction orders the collection of the transported cargo from the customer at the requested time. As an expected reaction, the cargo would be picked up and brought to the nearest hub. This behavior could be observed by a `CargoEntry` event at the hub.
- **Dispatch instruction**: Once a shipment lies in a hub, it needs to be loaded to a truck, in order to be transferred to the destination or possibly an intermediate hub. By issuing a dispatch instruction, the agent orders a shipment to be loaded on a truck, observable by a `CargoExit` event from the current hub onto the specified truck.
- **Movement instruction**: Truck movements between hubs are issued by movement instructions. A movement instruction specifies the truck, the origin and the

destination hub, as well as the time of departure. A movement can be observed by a sequence of events, starting with `Undocked` and `Departure` from the origin hub, followed by an `Arrival` and `Docked` event at the destination. Throughout the movement, the vehicle sends `LocationUpdate` and `TelematicsUpdate` events.

- **Delivery instruction**: A delivery instruction orders the final delivery to the customer. As an expected behavior the shipment would leave the hub, indicated by a `CargoExit` event from the destination hub.
- **Revocation instruction**: A revoke instruction orders the revocation of another instruction. This could be due to the cancelation of a movement or a rerouting of a shipment, where existing dispatch instructions for the shipment need to be revoked.
- **Postpone instruction**: A postpone instruction is issued for postponing an arbitrary instruction for the defined time.
- **Redirection instruction**: A redirection instruction is issued to change the destination of a movement instruction, possibly due to an unexpected event.

The set of all movement and dispatch instructions make up the schedule of the transportation network and provide a means for the software agent, to control the freight streams autonomously. Depending on the implementation of the routing policy, the schedule can vary in the degree of dynamism. Modeling a more static scheduling strategy can be done by predefining all (or the majority of) movement instructions according to a time table. In a dynamic setting, as it is proposed in this thesis, movements are ordered on demand, considering the utilization of the transportation network and can be planned with different planning horizons, reaching from only hours in a very dynamic setting, to multiple days for a more static setting. The majority of instructions does not have to be executed immediately after they have been issued, but rather at a certain time in the future. This is the reason why all instructions specify a scheduled execution time. Additionally, all instructions specify an estimated duration of execution. This duration is used in order to make predictions about the future state of the transportation network.

Note: Collection and delivery steps have only been modeled very rudimentary and would require much more detailed sensory data when considering all steps in the process of transportation. However, this thesis focuses around line haul operations with consolidated goods in between hubs. There is a big difference in requirements and algorithms used for local transportation (i.e. collections and deliveries) and line haul transportation [LW01]. Due to this difference, local transportation is a separate field of research. However, there have been tremendous efforts both by research and the industry to develop new effective ways of local freight distribution, since this part of the shipping process is – by distance – the most expensive, making up to 30% [GVdVV11, Goo05] of the overall transportation cost. The scenario covered by the thesis considers a transport order completed, once it reaches the destination hub, i.e. the closest hub to the delivery address.

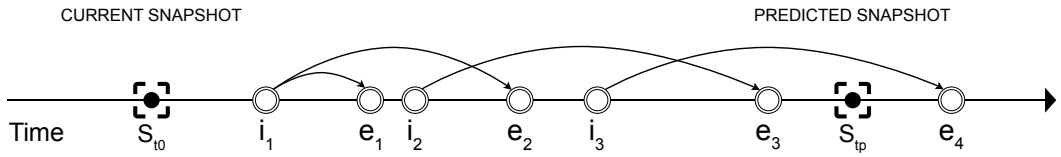


Figure 5.6: Predicting the future network state by applying expected events onto the current snapshot.

5.7 Future State Prediction

As it was mentioned before, the software agent optimizes the vehicle schedule based on snapshots of the transportation network. At each invocation cycle, the agent receives a snapshot and returns a set of instructions. One practical requirement for these cycles is, that – with the exception of incident handling – it is not reasonable to change the schedule at the last minute. This would leave no lead time for planning work shifts, coordinating with business partners or similar issues. Instead, the agent should preserve the imminent schedule and plan beyond, i.e. optimize based on a future state of the network. As an example, the schedule for the next day, denoted as the set of instructions $\{i \mid i \in I_t, t \in [0, 24]\}$, where the interval $\Delta_t = 1h$ should not be changed by the agent, thus also preserving the set of expected states $\{s_{t+i} \mid s_{t+i} \in S, i \in [0, 24]\}$. The agent would plan starting from $t = 25$ until reaching the planning horizon $t \leq t_{max}$. The default value for $t_{max} = 96h =$ four days.

These circumstances require a mechanism, that is capable of building snapshots of a predicted transportation network state S_{t_p} at a particular future time instant t_p. This can be done by taking all instructions $\{i \mid i \in I_t, t \in [0, t_p]\}$ that have been issued by the software agent to be executed before t_p and simulate these instructions, i.e. generate according events for each instruction. These simulated events can then be applied to a copy of the current transportation network state, effectively building an expected future network state. Figure 5.6 illustrates this principle. In the depicted example, three instructions $\{i_1, i_2, i_3\}$ are scheduled before the time of prediction t_p. These instructions cause the expected events $\{e_1, e_2, e_3, e_4\}$. The future state S_{t_p} is acquired by simulating all events that happen before t_p and applying the changes onto the copy of the current state S_{t_0}. In the particular case, these are $\{e_1, e_2, e_3\}$. Algorithm 1 (Appendix) defines the general procedure for acquiring a future snapshot. It creates all events according to the instructions, sorts them according to their time of emission and uses the default event handlers to apply changes onto the copy of the current snapshot. For reasons of brevity, the cases for collections and deliveries, as well as a number of non-essential other events were omitted in the algorithm.

The value of the parameters (t_p and t_{max}) have been chosen in consideration of the provided data set, without loss of generality. That is, choosing the parameters do not have implications for the framework, but only effect practical aspects of the transportation

process, such as work shift planning and provisioning of resources. Yet, t_p has to be chosen at a value that exceeds the run time of the optimization cycle, which for large networks could likely be hours.

The presented procedure is based on the simplifying assumption, that all events happen deterministically. Work on the stochastic prediction of events does exist [MTZ16, FFFM13], however this does not lie within the scope of this thesis.

5.8 Subsystem for Unexpected Event Handling

Unexpected events are a common problem for planning schedules in road transportation. They cause risks in terms of the adherence to deadlines or other service agreements and require fast decisions to be made by the operator or autonomous agent. Since the schedule consists of synchronized trips that follow timely constraints, the delay of one transportation can impact future transportations. Traditionally, with fixed schedules, this risk was minimized by planning trips with relatively high time buffers and cascading delays [Hal85, ZF10].

A number of external influences can disturb regular operations of logistics networks and cause such delays. Aside from traffic congestion and technical breakdowns, the weather has big influences on travel times, that sometimes cannot be predicted well. The framework aims at providing tools for early incident detection by using implicit information (i.e. event patterns) and explicit information (i.e. dedicated events) about the environment and provide a generic mechanism for context-aware incident handling.

5.8.1 Classification of Delays

Types of delays may roughly be grouped by the scope and potential impact on the schedule. This leads to the distinction of three essential cases:

- **Single-Vehicle Delay:** This form of delay concerns single vehicles and may be caused by technical issues of the vehicle. The impact on the schedule of single-vehicle delays is small. The transported shipments might miss their connection transportation trips and vehicles might not be able to go on their next scheduled trip.
- **Single-Connection Delay:** This form of delay is limited to single connections, but has a bigger impact on future schedule, as multiple movements might be affected. One major reason for single-connection delays is traffic congestion.
- **Multi-Connection Delay:** Finally, delays concerning multiple connections have the biggest impact on the future schedule, as they affect the largest number of movements and spread across wide geographical regions. Bad weather is one example for such types of delays.

In consideration of the classification provided above, the framework is capable of handling incidents from each of the three types. They are grouped by ascending scope and expected impact on the global network:

- **Technical breakdown**: Both minor and sever technical issues on vehicles are handled by the framework. The breakdowns are signaled by dedicated `Telematics` events, that can transmit detailed data about the technical state of the vehicle. In practical terms, minor and sever issues could comply with changing a flat tire up to requiring mobile mechanical service, but in both of the considered cases, the vehicle is eventually able to recover and continue its service. Complete breakdowns of long haul vehicles are extremely rare and for this reason have been excluded from considerations regarding the handling of incidents[5].

- **Traffic congestion**: Traffic delays are also considered by the framework. Depending on the duration, a congestion may not only have an immediate impact on the current movements on the connection, but also delay planned movements from the near future. In the model, congestions change the estimated travel time for connections in the transportation network. The traffic data is fetched from Google Distance Matrix API and propagated via the `TravelTimeUpdate` event. For the evaluation, varying connection times have been simulated by the environment simulator.

- **Poor weather conditions**: Unexpected changes in the weather cause similar effects to traffic congestion, but are not restricted to single connections. These events were modeled analogous to congestion, by modifying estimated travel times. For weather warnings, the Weather Company Data for IBM Bluemix API service was used.

Note: Traffic congestion in line haul can often be avoided by scheduling movements in times of low traffic activity, such as at night, but is not always possible (e.g. for same-day delivery). For this reason, according to the evaluated dataset, most of the recorded movements happened after 6:00 pm.

5.8.2 Handling Delays in Transportation Networks

If a vehicle is delayed, two separate problems arise. One is concerned with the vehicle itself, and the other one is concerned with the transported cargo. A delay becomes problematic with regard to the vehicle, if it is scheduled for another trip in the near future and is no longer able to perform it on time. Similarly, delays might cause shipments to miss the scheduled connection trips. In both cases, rescheduling with varying degrees of modifications to the existing schedule is required. The complete action and decision sequence for incident handling is depicted in the diagram in Figure 5.7. It illustrates the

[5]The information was acquired in a series of interviews with coworkers from DB Schenker.

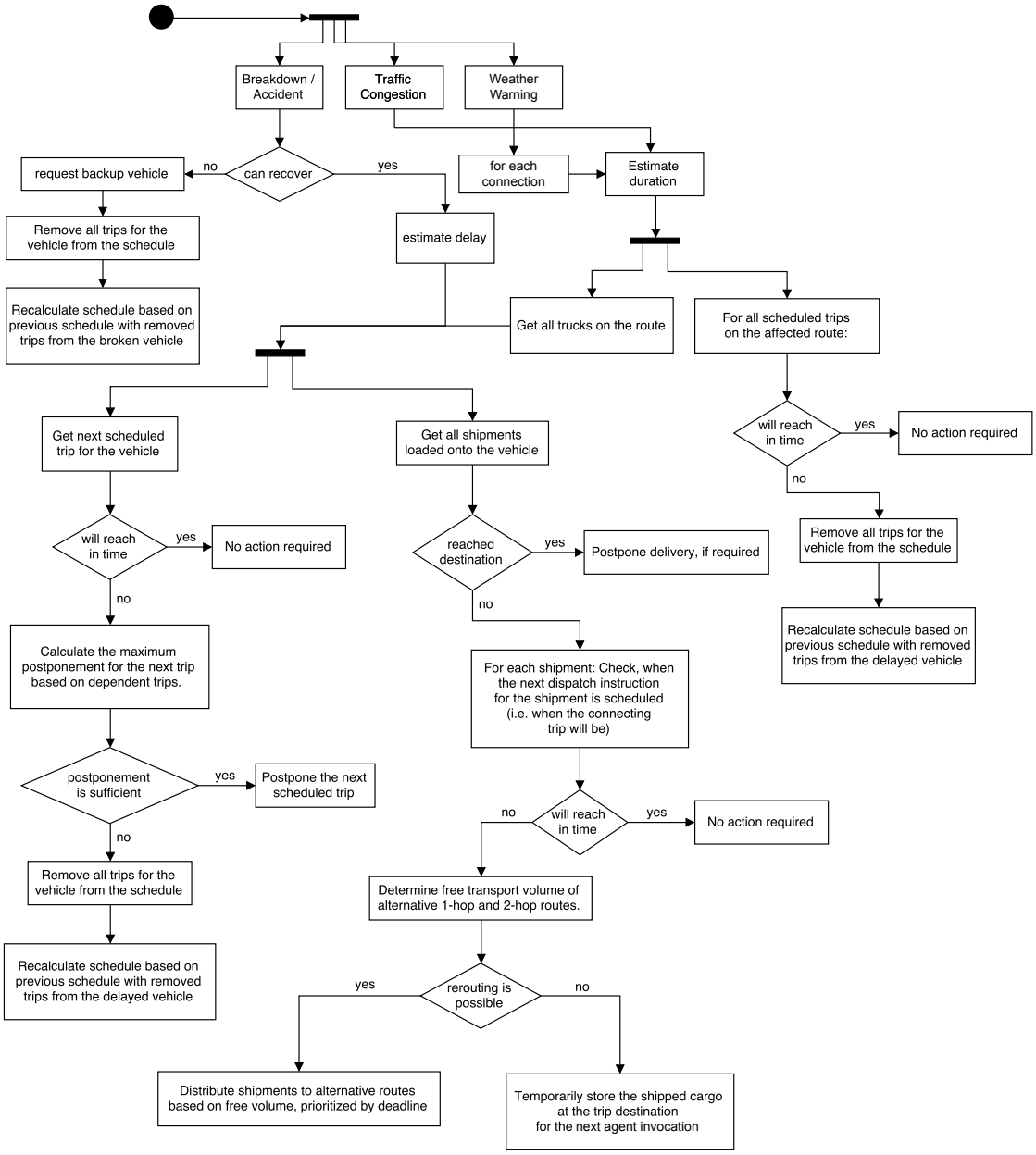


Figure 5.7: Action and decision sequence for handling unexpected delays of vehicles and connections.

process for context-aware incident handling and and extends the process of Ngai et al. [NLW+12]. The sequence deals with both the vehicle and the shipment aspects of delays, which can be done in parallel.

5.8.3 Rerouting Cargo

As it can be seen in the diagram, in some cases it is sufficient to delay subsequent instructions in order to get back to the desired operational state of the transportation network. However, certain branches of the decision process require a rerouting of cargo. For example, if a movement is delayed and the loaded shipments can no longer reach its connections, a search for alternative routes based on the existing schedule is conducted. For this reason, a rerouting algorithm was implemented that considers all 2-hop rerouting options for existing movements, preferring movements with earlier arrival dates.

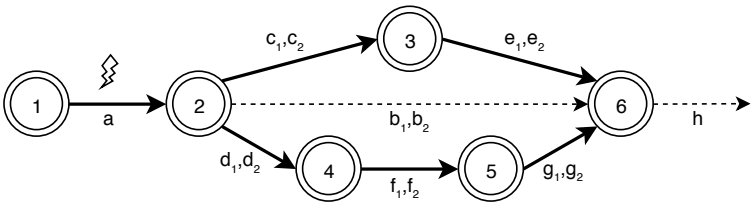


Figure 5.8: Available connections for rerouting cargo after a delay.

Scenario S_1

a	(0,5,8)
b_1	(4,7,10)
b_2	(12,15,2)
c_1	(9,12,2)
d_1	(6,7,8)
e_1	(13,14,2)
f_1	(8,10,8)
g_1	(11,12,8)
h	(10,12,10)

Scenario S_2

a	(0,5,8)
b_1	(4,7,10)
b_2	(12,15,2)
c_1	(9,12,3)
d_1	(6,7,8)
e_1	(13,14,3)
f_1	(8,10,8)
g_1	(11,12,5)
h	(10,12,10)

Scenario S_3

a	(0,5,8)
b_1	(4,7,10)
b_2	(12,15,0)
c_1	(9,12,3)
d_1	(6,7,3)
e_1	(13,14,3)
f_1	(8,10,3)
g_1	(11,12,3)
h	(10,12,10)

Table 5.2: Distinct shipment rerouting scenarios Pattern: *(arrival, departure, unoccupied volume)*

Example scenarios

Three scenarios have been chosen to illustrate the principles of the proposed rerouting algorithm. Figure 5.8 provides a set of movements $\{a, b_i, c_i, d_i, e_i, f_i, g_i, h \mid i \in \{1, 2\}\}$ through a subset of hubs in a transportation network. In all of the three scenarios provided in Table 5.2, the originally planned route $\{a, b_1\}$ is not feasible, due to a delay

during the first movement a. The notation for a movement in the scenarios corresponds to the pattern *(arrival, departure, unoccupied volume)*.

- **Scenario S_1:** In this scenario, 8 volume units from movement a need to be rerouted. For this, three options are available: The 1-hop route over hub 3 (3 unoccupied volume units, arrival at $t + 14$), the 2-hop route over hubs 4 and 5 (8 unoccupied volume units, arrival at $t + 12$), as well as a direct connection via b_2 (2 unoccupied volume units, arrival at $t + 15$). In this case, because the 2-hop option offers enough unoccupied volume and has the earliest arrival, the entire volume is rerouted via that route.

  Result: $a(8) \rightarrow d_1(8) \rightarrow f_1(8) \rightarrow g_1(8)$

- **Scenario S_2:** This scenario is similar to the first scenario, except that the unoccupied volume for the 2-hop option is not sufficient to cover the full volume to be rerouted, as the link with the minimum unoccupied volume g_2 determines the unoccupied volume for the entire route. For this reason, the remaining volume is rerouted to the second fastest option via hub 3.

  Result: $a(8) \rightarrow \begin{matrix} c_1(3) \rightarrow e_1(3) \\ d_1(5) \rightarrow f_1(5) \end{matrix} \rightarrow g_1(5)$

- **Scenario S_3:** Finally, the third scenario covers a case, where the alternative routes do not offer enough unoccupied volume to cover the full volume to be rerouted. In this case, the remaining volume is temporarily stored at the intermediate hub 2.

  Result: $a(8) \rightarrow \begin{matrix} c_1(3) \rightarrow e_1(3) \\ d_1(3) \rightarrow f_1(3) \end{matrix} \rightarrow g_1(3)$

CHAPTER 6

Dynamic Transshipment with Time Constraints

In this chapter the second of two contributions, a mathematical optimization model for building a dynamic vehicle schedule will be introduced. It extends the Dynamic Transshipment Problem and additionally considers deadline penalties, loading gate restrictions, and recalculations based on previous solutions. In detail, this includes models for two separate optimization problems. The first optimization problem (Section 6.2) determines the demand of trucks for each connection and time, given all transport orders. The second optimization problem (Section 6.3) uses the output of the first model and assigns vehicles to the demanded transportation routes. The result of a serial execution of both models is a dynamic truck schedule.

6.1 Overview

The proposed optimization problem for optimizing freight streams implements a software agent policy $\Pi_t(s, a)$, and will further be referred to as the routing policy. In particular, this routing policy, where s denotes the compound state of the transportation network and a denotes all instructions that are required to operate a transportation network (e.g., move vehicles, dispatch shipments to vehicles) optimizes costs with respect to a cost parameter vector c. In the routing policy definition, $s = \{O, S_H, S_T, S_C, c\}$ comprises the set of pending transport orders O, all hub states S_H, vehicle states S_T, and connection states S_C, as well as the cost parameter vector c. This vector $c = (c_d, c_t, c_s, c_{dl})$ provides parameters for travel costs c_d (per distance unit), turnover and storage costs c_t and c_s (per weight unit), and deadline penalty costs c_{dl} (per weight and time unit).

6.2 Dynamic Transshipment with Time Constraints in a Finite Planning Horizon

The proposed problem definition resembles the Dynamic Transshipment Problem, introduced by Herer and Tzur [HT01], where for each time period t, a deterministic demand of order volumes is issued. The definition considers additional constraints: Each order, in addition to specifying a dedicated pickup time, specifies a delivery date. If the calculated route cannot adhere to the delivery date, penalties are charged. In addition, loading gate restrictions need to assure that the amount of vehicles scheduled to simultaneously arrive at a hub does not exceed the number of loading gates, thus avoiding waiting times for terminal processing. For recalculations based on previous solutions, a means of providing previous solutions as input is required. This also needs to consider the dynamism of the setting, as trucks may already be on the road, instead of being a resource that is available at a hub, as it was traditionally assumed for similar problem definitions [Khu15]. In the next two sections, the optimization problem will be formally defined.

6.2.1 System Model

According to the data model presented in the previous chapter, a number of static and dynamic properties of the involved entities can be used as inputs for optimization. Table 6.1 lists all variables and identifiers that have been used in the problem definition. For a hub $h \in H$, this includes the storage capacity $cap(h)$ and the number of loading gates $lg(h)$. Trucks are constrained by their overall capacity cap_{tr}. Note that the problem definition assumes a homogeneous fleet, i.e., the capacity of each truck is equal. The two matrices $K^a_{t,h}$ and $K^d_{t,h}$ define a priori known truck movements, i.e., movements from previous solutions. Each departure $K^d_{t',h'}$ decrements the number of available trucks $\bar{A}_{t',h'}$ at hub h' at time t' (with an analogous increment for each arrival $K^a_{t',h'}$). This distinction is made, since movements that are currently in transit only specify an arrival, as the departure lies in the past. Similar to the classical Transshipment Problem, a shipment matrix M_{t,s,h_1,h_2} specifies the ordered volume between two hubs h_1 and h_2 at each time $t \in T$, resembling all transport orders . This matrix encodes both the pickup and delivery hub (as dimensions), as well as the pickup time (another dimension), and the volume of a shipment (matrix value). However, there is an additional dimension for s in M, the shipment. This is because of one fundamental modeling problem of a multi-shipment scenario: Assuming that a shipment *s1* specified 20 volume units to be shipped from hub A to B and a second shipment *s2*, that specified 20 volume units from B to A without the dimensional separation for the shipments, the optimal solution would be to not make any movement at all, as the desired equilibrium state for supply and demand would already be satisfied. The separate dimension prevents this behavior, but adds computational complexity, especially for large dimension spaces, as it is the case for shipments. As a countermeasure to reduce the amount of shipments, shipments with the same origin, destination, as well as similar pickup and delivery times could be combined into a single shipment. However, this would create larger atomic units for transportation,

which in turn influences the quality of the solution.

In addition to the shipment matrix, $dl(s)$ specifies the deadline of a shipment. Missing the deadline causes the deadline penalty c_{dl} to be added to the costs for each additional time unit. The time $t \in T$ in the model is defined in discrete intervals, limited by the planning horizon t_{max}, i.e., $T = [0, .., t_{max}]$. $\tau(h_1, h_2)$ defines the travel time from h_1 to h_2, based on current and predicted travel times for each connection. Another constant θ is used for the turnover time, combining unloading and loading. In order to leverage existing movements from previous solutions, $occ(t, h_1, h_2)$ specifies the occupied volume for existing movements. If the occupied volume for a connection (t, h_1, h_2) does not equal the capacity of all trucks scheduled for this connection, unoccupied volume from previous solutions is available. Dispatching shipments to these movements will not require new vehicles to be provisioned, thus improving vehicle utilization.

A solution of the problem consists of four decision matrices $\bar{V}$, $\bar{V}^H$, $\bar{A}$, and $\bar{R}$ (as a notational convention for easier distinction between input matrices and matrices that are decision variables, decision matrices are marked with a top bar, e.g., $\bar{V}$). The first matrix $\bar{V}$ specifies the volume flow of shipments through the network. It defines the amount of shipped volume between each two hubs at each time t. This volume flow, together with the volume from previous solutions results in the stored volume at each hub, denoted by $\bar{V}^H_{t,h}$. It also results in the number of trucks $\bar{R}_{t,h_1,h_2}$ required to ship the volume for each connection and time. Finally, the number of available trucks $\bar{A}_{t,h}$ at each hub is a result of $\bar{R}$, K^a and K^d. The main output of a solution are the matrices $\bar{V}_{t,s,h_1,h_2}$ and $\bar{R}_{t,h}$. Once a vehicle assignment for $\bar{R}$ is found by the second optimization problem (discussed in Section 6.3), all shipments defined in $\bar{V}$ can be dispatched to their according vehicles.

6.2.2 Optimization Problem

The goal of the optimization problem is to minimize costs while processing the transport orders. The objective function, depicted in Equation 6.1, minimizes the sum of all travel costs and deadline penalties that become due for late shipments. The function $\mu(s, h_1, h_2)$ (Equation 6.2) denotes the interval between the deadline $dl(s)$ of a shipment s and the planning horizon t_{max}. The travel time $\tau(h_1, h_2)$ between two hubs h_1 and h_2 is subtracted from the deadline, since it is necessary to start a movement at $dl(s) - \tau(h_1, h_2)$ in order to not miss the deadline. For each time step $t > dl(s)$ where the shipment has not arrived at the destination hub (i.e. $\bar{V}_{t,s,h1,h2} > 0, \forall h1, h2 \in H$), the penalty rate c_{dl} is added to the overall costs.

$$\min_{x \in \mathbb{Z}^+} \underbrace{\sum_{h_1,h2 \in H} \sum_{t \in T} \tau(h_1, h_2) \cdot \bar{R}_{t,h_1,h_2} \cdot c_{km}}_{\text{travel costs}} + \underbrace{\sum_{h_1,h_2 \in H} \sum_{s \in S} \sum_{t \ \in \mu(s,h_1,h_2)} min(1, \bar{V}_{t,s,h_1,h_2}) \cdot c_{dl}}_{\text{deadline penalties}} \tag{6.1}$$

Table 6.1: Variable Description: Dynamic Transshipment Problem with Time Constraints in a Finite Planning Horizon

Variable Name	Description
H	Set of hubs
$lg(h)$	Number of loading gates at hub h
$cap(h)$	Maximum storage capacity of hub h
S	Set of shipments
M_{t,s,h_1,h_2}	Ordered movement volume for shipment s from origin h_1 to destination h_2, starting at time t
$K^a_{t,h}$	Initial number of arriving trucks at hub h for each time t
$K^d_{t,h}$	Initial number of departing trucks for hub h for each time t
c_{km}	Vehicle costs per km
c_{dl}	Costs for missed deadline per time unit after deadline
cap_{tr}	Capacity of a truck
$dl(s)$	Deadline of a shipment s
$\tau(h_1, h_2)$	Travel time from hub h_1 to hub h_2
t_{max}	Planning horizon (i.e. number of considered time steps)
$T = [0, .., t_{max}]$	Discrete notion of time
$\alpha_t(s) : S \longrightarrow T$	Initial arrival of shipment s
$\alpha_h(s) : S \longrightarrow H$	Origin hub of shipment s
θ	Turnover time (i.e. unloading and loading)
$occ(t, h_1, h_2)$	Preoccupied volume from hub h_1 to hub h_2 at time t
$\bar{V}_{t,s,h_1,h_2}$	Shipped volume for shipment s at time t from hub h_1 to hub h_2
$\bar{V}^H_{t,h}$	Stored shipment volume of hub h at time t
$\bar{A}_{t,h}$	Available trucks at hub h and time t
$\bar{R}_{t,h_1,h_2}$	Number of traveling trucks from hub $h1$ to hub $h2$ at time t

Variables below the horizontal rule are decision variables

$$\mu(s, h_1, h_2) = [max(0, dl(s) - \tau(h_1, h_2)), t_{max}] \tag{6.2}$$

The decision variables $\bar{V}_{t,s,h_1,h_2}$ and $\bar{R}_{t,h_1,h_2}$ denote the transported volume and the necessary number of vehicles. Equation 6.3 describes the relationship that needs to be satisfied by both matrices. The first term aggregates the overall volume to be shipped between two hubs h_1 and h_2 at time t. It consists of the shipped volume from $\bar{V}$ and the occupied volume $occ(t, h_1, h_2)$. It is the volume of existing movements (e.g., scheduled in previous optimization cycles). This helps to better utilize existing movements before assigning additional vehicles to the route. The second term describes the provisioned vehicle capacity. It determines the final number of vehicles that is demanded for transporting the calculated volume from $\bar{V}$.

$$\forall h_1, h_2 \in H, t \in T : \sum_{s \in S} \bar{V}_{t,s,h_1,h_2} + occ(t, h_1, h_2) \leq \bar{R}_{t,h_1,h_2} \cdot cap_{tr} \tag{6.3}$$

Equation 6.4 ensures the correct flow of shipments through the transportation network. Generally speaking, it ensures that for each hub, over time, the sum of incoming shipment volume subtracted by the sum of outgoing shipment volume (i.e., the left member of the equation) equals the supply and demand, defined in M (the right member of the equation). This assures that for supply nodes, the balance of incoming and outgoing volume equals the negative supply (as outgoing volume exceeds incoming volume), the demand nodes volume balance equals the supply (as, according to M the demand of a shipment always equals the supply) and the volume balance for intermediate hubs always equals zero (i.e., incoming volume is the same as outgoing volume). Note that a node can be a supply node, intermediate node and demand node for different shipments at the same time.

$$\begin{aligned} \forall h_1 \in H, s \in S : \\ \sum_{h_2 \in H} \sum_{t \in T} \bar{V}_{t,s,h_1,h_2} \quad - \sum_{h_3 \in H} \sum_{t \in T} \bar{V}_{t,s,h_3,h_1} \\ = \sum_{h_4 \in H} \sum_{t \in T} M_{t,s,h_1,h_4} \quad - \sum_{h_5 \in H} \sum_{t \in T} M_{t,s,h_5,h_1} \end{aligned} \tag{6.4}$$

In Equation 6.5, the hub storage volume matrix $\bar{V}^H$ is defined. The stored volume $\bar{V}^H_{t,h_1}$ of a hub h_1 comprises the supply defined by $\sum_{h_2 \in H} \sum_{s \in S} M_{t,s,h_1,h_2}$, the incoming or existing volume, denoted as $bal_V(t, h)$ and defined in Equation 6.7, subtracted by the outgoing volume $\sum_{h_2 \in H} \sum_{s \in S} \bar{V}_{t,s,h_1,h_2}$. $bal_V(t, h)$ is defined for three distinct cases. The first case ($t = 0$) considers no additional volume. In the second case ($t > 0 \wedge t - \tau(h_2, h_1) < 0$) the volume of the previous time step is added. This case covers situations, where the transport from $h2$ to $h1$ has not yet arrived (i.e., $t - \tau(h_2, h_1) < 0$). The third case ($t > 0 \wedge \tau(h_2, h_1) \geq 0$) additionally considers incoming volume from routes that departed from h_2 at $\tau(h2, h1)$.

$$\forall h_1 \in H, t \in T : V^H_{t,h_1} = \sum_{h_2 \in H} \sum_{s \in S} M_{t,s,h_1,h_2} - \bar{V}_{t,s,h_1,h_2} + bal_V(t, h_1) \tag{6.5}$$

$$\forall h_1 \in H, t \in T : \bar{A}_{t,h_1} = K^a_{t,h_1} - \sum_{h_2 \in H} \bar{R}_{t,h_1,h_2} + bal_R(t, h_1, h_2) \tag{6.6}$$

The number of available vehicles per hub are calculated in Equation 6.6. Similar to Equation 6.5, it consists of three terms that represent initial vehicles (K^a), vehicles from the last time step or from other hubs (bal_R) and outgoing vehicles $\sum_{h_2 \in H} \bar{R}_{t,h_1,h_2}$. Equation 6.8 is defined analogous to Equation 6.7, where $\phi(t, h_1, h_2)$ – Equation (6.9) – denotes the difference between available vehicles from the previous time step $\bar{A}_{(t-1),h_1}$ and predefined departures from K^d.

$$bal_V(t, h_1) = \begin{cases} 0 & \text{for } t = 0 \\ \bar{V}^H_{(t-1),h_1} & \text{for } t > 0 \wedge t - \tau(h_2, h_1) < 0 \\ \bar{V}^H_{(t-1),h_1} + \sum_{h_2 \in H, s \in S} \bar{V}_{t-\tau(h_2,h_1),s,h_2,h_1} & \text{otherwise} \end{cases} \tag{6.7}$$

$$bal_R(t, h_1, h_2) = \begin{cases} 0 & \text{for } t = 0 \\ \phi(t, h_1, h_2) & \text{for } t > 0 \wedge t - \tau(h_2, h_1) < 0 \\ \phi(t, h_1, h_2) + \bar{R}_{t-\tau(h_2,h_1),h_2,h_1} & \text{otherwise} \end{cases} \tag{6.8}$$

$$\phi(t, h_1, h_2) = \bar{A}_{(t-1),h_1} - K^d_{t,h_1,h_2} \tag{6.9}$$

Equations 6.10 and 6.11 restrict the flow of volume through the network by limiting hub capacities $cap(h)$ and the number of loading gates $lg(h)$. Equation 6.10 ensures that the hub capacity is exceeded at no time. The subsequent Equation 6.11 restricts the number of departing trucks, i.e., $\sum_{h_2 \in H} \bar{R}_{t-\tau(h_2,h_1),h_2,h_1}$, which simultaneously arrive at h_1 to the number of loading gates $lg(h_1)$ of h_1.

$$\forall h \in H, t \in T : cap(h) \geq \bar{V}^H_{t,h} \tag{6.10}$$

$\forall h_1 \in H, t \in T :$

$$lg(h) \geq \begin{cases} \sum_{h_2 \in H} \bar{R}_{t-\tau(h_2,h_1),h_2,h_1} & \text{for } t - \tau(h_2, h_1) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

The model further assumes atomicity of transport orders, i.e., the volume of one transport order cannot be split up and take separate routes, but has to traverse the network as a whole. This constraint is defined in Equation 6.12.

$\forall t \in T, s \in S, h_1, h_2 \in H, :$

$$\bar{V}_{t,s,h_1,h_2} = \begin{cases} \sum_{t' \in T} \sum_{h_3,h_4 \in H} M_{t',s,h_3,h_4} & \text{for } \bar{V}_{t,s,h_1,h_2} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

The optimization model also considers a turnover time θ for shipments, as loading and unloading requires time. For this reason, if a shipment arrives at an intermediate hub at time t, it cannot immediately be transferred to the next hub, but has to be on standby at the hub for the interval $[t, t+\theta]$. This restriction is defined in Equation 6.13. The same principle applies to waiting times for initially arriving shipments. Equation 6.14 assures that shipments are not being transferred for θ time steps after their initial arrival at $\alpha_t(s)$.

$$\forall s \in S, t \in T, h_1, h_2 \in H : \begin{cases} \sum_{h_3,h_4 \in H} \bar{V}_{t+\tau(h_1,h_2)+\theta,s,h_3,h_4} = 0 & \text{for } \bar{V}_{t,s,h_1,h_2} > 0 \\ \top & \text{otherwise} \end{cases} \tag{6.13}$$

$$\forall s \in S : \sum_{h_1,h_2 \in H} \sum_{t \in [\alpha_t(s), \alpha_t(s)+\theta]} \bar{V}_{t,s,h_1,h_2} = 0 \tag{6.14}$$

6.3 Vehicle Assignment

Solutions for the model discussed in the previous section include four matrices that describe the freight streams in the transportation network, most notably the resulting matrix $\bar{V}$, which defines the routing for each shipment through the network and the matrix $\bar{R}$, which holds the required number of trucks for each connection and time step. This matrix will serve as input for the second optimization problem proposed in the thesis and discussed in this section.

6.3.1 System Model

In order to build a dynamic schedule, concrete vehicles need to be assigned to the demand $\bar{R}$, resulting from the first optimization problem. The optimization problem discussed below provides a general approach and allows for an extension with additional constraints. For example, a vehicle assignment might consider working shifts or round trip times for human drivers, but could also define more loose constraints for an autonomous fleet.

Table 6.2 provides an overview of all variables used in the problem definition. The three sets H, M and Tr represent the involved entities, where M denotes the movements resulting from the solution of the Transshipment Problem and Tr denotes the set of all trucks. Similar to the first problem definition, H represents all hubs in the network. The lookup function $h(tr)$ returns the home hub for each truck. Additional lookup functions include $\alpha_h(m)$ and $\omega_h(m)$ for origin and arrival hubs of movements, as well as $\alpha_t(m)$ and $\omega_t(m)$ for departure and arrival times. $\delta(m)$ denotes the number of required trucks for the movement m.

Assigning vehicles to movements requires knowledge of which vehicle is available at which hub at what time. In order to track a vehicle's location in the network, the two decision

Table 6.2: Variable Description: Vehicle Assignment Problem

Variable Name	Description
H	Set of hubs
M	Set of movements
Tr	Set of trucks
$h(tr)$	Home hub of truck tr
$A_{t,tr,h} \in \{0,1\}$	A priori known arrivals of truck tr at hub h and time t
$I_{t_d,t_a,h_1,h_2,tr} \in \{0,1\}$	A priori known schedule for truck tr, with departure t_d at hub h_1 and arrival t_a at hub h_2
t_{max}	Planning horizon (i.e. number of considered time steps)
$T = [0,..,t_{max}]$	Discrete notion of time in steps
$\alpha_t(m)$	Departure time of movement m
$\alpha_h(m)$	Origin hub of movement m
$\omega_t(m)$	Arrival time of movement m
$\omega_h(m)$	Destination hub of movement m
$\delta(m)$	Demanded number of trucks for movement m
$\bar{S}_{t,tr,h} \in \{0,1\}$	Standby status for truck tr at hub h and time t
$\bar{R}_{tr,m} \in \{0,1\}$	Assignment for truck tr to movement m

Variables below the horizontal rule are decision variables

matrices $\bar{S}$ and $\bar{R}$ are required. A vehicle can either be available at a hub ($\bar{S}_{t,tr,h} = 1$), or in transit ($\bar{R}_{tr,m} = 1$), but never both. $\bar{S}$ and $\bar{R}$ partly depend on the a priori known schedule for vehicles, i.e. movements from previous solutions. Similar to the previous problem definition, where arrivals and departures where separated, a distinction is made for movements that are in transit and movements that are scheduled in the future. For this reason, $A_{t,tr,h}$ specifies the arrival of each truck at each hub and time for movements in transit. Similarly, $I_{t_d,t_a,h_1,h_2,tr}$ is defined for future movements, additionally specifying the departure time t_d and the origin hub h_1.

6.3.2 Optimization Problem

The vehicle assignment problem attempts to find a mapping between vehicles and vehicle demand in order to build a dynamic schedule for vehicles. The objective function depicted in Equation 6.15 maximizes the standby time of trucks at their home hub. This is a desirable solution property, as it enables an easier management of the vehicle fleet and personnel [Gud12]. If round trip times are short, working schedules for drivers can be planned more flexibly. However, the objective might be adapted for autonomous fleets. The constraint defined in Equation 6.16 assures that the number of assignments equals the number of required trucks for each movement.

$$\underset{x \in \mathbb{Z}^+}{max} \sum_{t \in T} \sum_{tr \in Tr} \sum_{m \in M} \bar{S}_{t,tr,t(h)} \tag{6.15}$$

$$\forall m \in M : \sum_{tr \in Tr} \bar{R}_{tr,m} = \delta(m) \tag{6.16}$$

The standby status $\bar{S}_{t,tr,h} \in \{0,1\}$ of a truck tr is defined in Equation 6.17. It uses three indicators: The status of the previous time step $\bar{S}_{t-1}$, the a priori schedule from previous optimization cycles, comprising of scheduled arrivals A and movements I, and newly scheduled movements $\bar{R}$. The equation distinguishes between two cases, depending on the value for t. Informally the term evaluated for $t > 0$ returns incoming movements arriving at time t (denoted by σ_i), subtracted by outgoing movements leaving at time t (denoted by σ_o). In the case of $t = 0$, there cannot be any incoming movements. There is also no previous status $\bar{S}_{t-1}$.

$$\begin{aligned} &\forall tr \in Tr, h \in H, t \in T : \\ &\quad \bar{S}_{t,tr,h} = A_{t,tr,h} + \sum_{m \in arr(t,h)} \bar{R}_{tr,m} - \sum_{m \in dep(t,h)} \bar{R}_{tr,m} \\ &\qquad + \begin{cases} \sigma_i(H, P_t, t, h, tr) - \sigma_o(H, F_t, t, h, tr) + \bar{S}_{(t-1),tr,h} & \text{for } t > 0 \\ -\sigma_o(H, T, t, h, tr) & \text{otherwise} \end{cases} \end{aligned} \tag{6.17}$$

The functions $dep(t,h)$ (Equation 6.18), $arr(t,h)$ (Equation 6.19) and $dur(t)$ (Equation 6.20) are filters that return specific subsets of the set of all movements M. $dep(t,h)$ filters for all movements m that depart from hub h at time t. Similarly, $arr(t,h)$ selects all movements that arrive at hub h and time t. $dur(t)$ returns all active movements m (i.e., $\alpha(m) \leq t \leq \omega(m)$) at time t.

$$dep(t,h) = \{m \mid m \in M,\, \alpha_t(m) = t,\, \alpha_h(m) = h\} \tag{6.18}$$

$$arr(t,h) = \{m \mid m \in M,\, \omega_t(m) = t,\, \omega_h(m) = h\} \tag{6.19}$$

$$dur(t) = \{m \mid m \in M,\, \alpha_t(m) \leq t,\, \omega_t(m) \geq t\} \tag{6.20}$$

The function $\sigma_i : \{H\} \times \{T\} \times H \times T \times Tr \longrightarrow \{0,1\}$, defined in Equation 6.21 indicates, whether there is any known incoming movement (from the initial schedule $\bar{I}$) to the specified hub within the specified time interval. Analogous, the function σ_o indicates outgoing movements within a specified time interval. For this purpose, the sets P_{t_0}

(Equation 6.24) and F_{t_0} (Equation 6.25) are defined as all past and all future time steps, relative to t_0. Note that the invariant depicted in Equation 6.23 must hold. It ensures that a vehicle is either in standby (only at a single hub), or on a movement (known or calculated), but never more than one thing at the same time.

$$\sigma_i(H, T, t, h, tr) = \sum_{o \in H} \sum_{d \in T} I_{d,t,o,h,tr} \tag{6.21}$$

$$\sigma_o(H, T, t, h, tr) = \sum_{d \in H} \sum_{a \in T} I_{t,a,h,d,tr} \tag{6.22}$$

$$\forall t \in T, tr \in Tr : \\ \sum_{m \in dur(t)} \bar{R}_{tr,m} + \sum_{h \in H} (\bar{S}_{t,tr,h} + \sigma_i(H, P_t, t, h, tr) + \sigma_o(H, F_t, t, h, tr)) = 1 \tag{6.23}$$

$$P_{t_0} = \{t \mid t \in T,\, t < t_0\} \tag{6.24}$$

$$F_{t_0} = \{t \mid t \in T,\, t > t_0\} \tag{6.25}$$

6.4 Conclusion

This chapter introduced an implementation of a routing policy $\Pi_t(s, a)$ based on mathematical optimization for creating dynamic vehicle schedules based on transportation demand. Two mathematical models were used for scheduling the vehicles. The first model, designed after the Transshipment Problem, allocates transportation volume to routes, adding constraints for deadlines and turnover times and calculating the required number of vehicles for each route. The second optimization model assigns concrete vehicles to the provisioned volume, effectively building a dynamic vehicle schedule. In the next chapter, this routing policy together with the underlaying framework and the subsystem for incident handling will be evaluated. The evaluation will facilitate real-world data sets for building a realistic setting and and assure the practicality of the chosen approach.

CHAPTER 7

Evaluation

In this chapter, we discuss the performance of our proposed freight stream optimization framework. During the evaluation, we focus on two main aspects of the framework, namely dynamic vehicle routing and automatic incident handling. Hence, our evaluation will be two-fold. First, we focus on the performance of the routing policy $\Pi_t(s, a)$, which is responsible for the dynamic dispatching of shipments and the routing of vehicles. We test the performance by simulating two weeks of transportation activity and comparing the resulting schedule to a fixed schedule. To this end, we facilitate data from the Austrian logistics company DB Schenker, which provides us with detailed transportation data, such as transport orders and movements, which are based on a fixed schedule. With the simulations, we aim to show the superiority of dynamic scheduling compared to a fixed schedule in terms of vehicle utilization, number of necessary movements, and overall driven kilometers. For these particular experiments, we do not consider incidents, such as vehicle breakdowns or traffic delay. Incident handling will be subject to the second part of the evaluation, where we add unexpected events to our simulations to test the subsystem for incident handling. Again, we base these simulations on the data provided by DB Schenker and simulate different scenarios to observe implications on the performance of our routing policy and framework.

7.1 Performance of Routing Policy

To test the performance of the proposed routing policy, we analyze the available data from DB Schenker and derive simulations of transportation activity. To that end, we built an event simulator that is capable of mimicking certain aspects of the environment, i.e., the transportation network, by issuing according events. This includes incoming transport orders, parcel entry and exit scans, vehicle docking and undocking, as well as GPS signals and other telemetric data from vehicles in transit. We refer to Figure 5.4 in Chapter 5 for the detailed event model that describes the framework's view on the

Table 7.1: Overview of the Datasets, each Covering a Week of Network Activity

	$\|O\|$	O_w	$\|M\|$	M_{ut}	o_w*	m_w*	$\frac{o}{m}$*
Set Q1	10468	2183t	643	36%	(1, 7263, 210)	(1, 19267, 3425)	(1, 105, 13)
Set Q4	9440	2053t	549	39%	(1, 10180, 218)	(1, 167444, 3741)	(1, 144, 14)

* ... (min, max, avg)

environment. The data from DB Schenker does not include all data considered in our environment model, but is limited to transport orders, entry and exit scans as well as planned departures and arrivals of vehicles.

7.1.1 Transportation Data

In detail, the dataset consists of LTL transportation records from eleven Austrian branches during a timespan of two weeks, one in Q1 and one in Q4 of 2015. Each week comprises between 60,000 and 70,000 production orders, i.e. orders for loading and unloading cargo and vehicle movements, and approximately 10,000 transport orders. We use this data to reveal freight streams, vehicle movements, and activity patterns in this section. For international orders in the dataset, which account for 60% of all recorded orders, we only consider the Austrian part of the transportation and adapt origins and destinations accordingly. Regarding capacity constraints of vehicles, the weight, and dimensions of shipments need to be considered. Unfortunately, the dataset does not include reliable information about cargo dimensions, which poses a challenge for comparing the results. In this section we explain, how we mitigated this problem by deriving weight-based capacity constraints from the data.

Data Characteristics

Table 7.1 lists the main characteristics for both datasets Set Q1 and Set Q4. In the table, $|O|$ and O_w refer to the number of transport orders and the overall weight of transport orders, $|M|$ refers to the number of vehicle movements, o_w and m_w are the minimum, maximum, and average transport order weights and transported weight per movement in kilogram, and $\frac{o}{m}$ provides the same metric for orders per movement. M_{ut} refers to the average utilization per movement under the same assumptions that we used for the simulations. Overall, the data shows seasonal fluctuations of the transported weight (6% gap) and number of movements (15% gap). The utilization of vehicles varies between 36% and 39%, i.e., typical vehicles were loaded at 36-39% to their capacity.

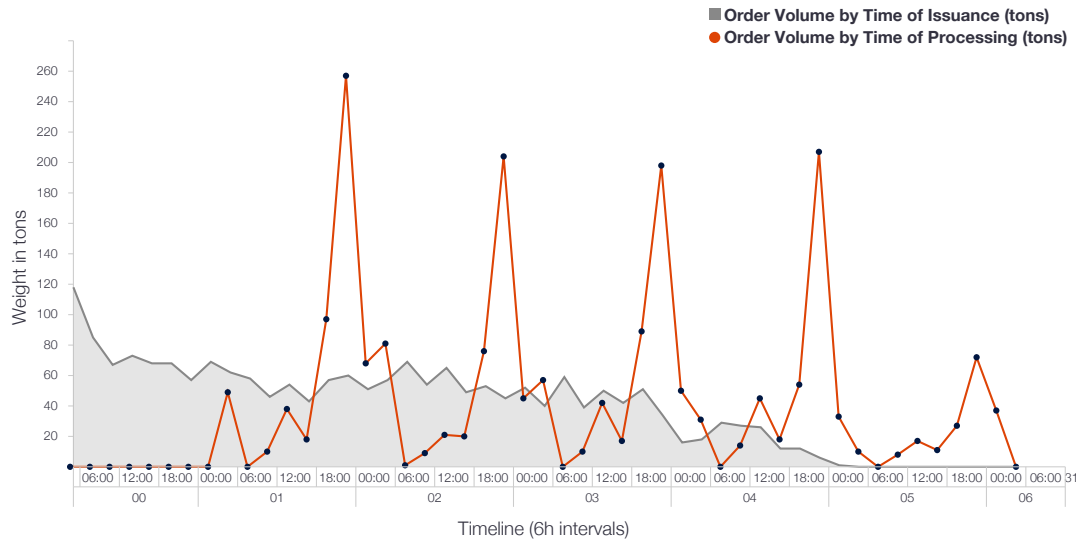


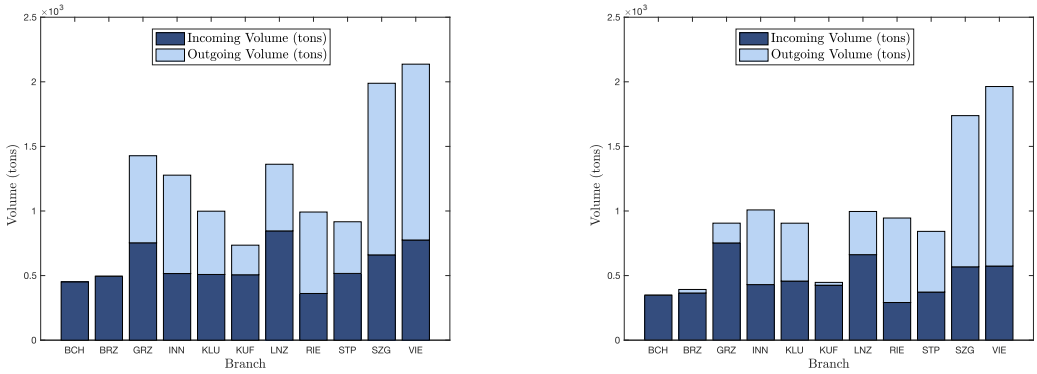
Figure 7.1: Timeline of Incoming vs. Processed Order Volume (Set Q4)

Transportation Demand

Figure 7.1 illustrates the pattern in which orders arrive in the system. Each data point in the graphic corresponds to the volume of 3h intervals within a work week – starting on Sunday with orders from the previous week – recorded in November of 2015 (Set Q4). Note that despite the word volume, in logistics, this term refers to weight in tons [Gud07]. The timeline shows that the number of incoming orders declines steadily throughout the week and more than 99.5% of orders are issued before Friday. The timeline also reveals the pattern in which these orders are processed, i.e., loaded and moved. Due to the fixed schedule, this pattern is very regular and has its peeks after rush hours, in order to avoid heavy traffic. Figures 7.2 a and 7.2 b provide a more detailed view into the distribution of order volumes to the eleven nodes in the network. Each graphic depicts the turnover volume of a dataset and breaks down the volume into incoming and outgoing volumes per node. The figures show the typical imbalances between incoming and outgoing transport volume, originating from the fact that some cities are dominant in production, whereas other cities are consumption-dominant. This fact has big implications on the utilization of vehicles, as they tend to have highly utilized trips to consumer destinations, but less utilized or even empty return journeys [Chr16]. Planning circular trips with multiple destinations is a possible solution to mitigate this problem.

Simulation Parameters and Assumptions

To compensate for missing data in the datasets, as well as computational resource limitations, we require further assumptions and restrictions in certain aspects, which we describe in the following.



(a) Set Q1 (b) Set Q4

Figure 7.2: Turnover Volume: Incoming and Outgoing Volume per Hub

Adaptions to the model. Although being mostly homogeneous, DB Schenker does not solely operate its own vehicle fleet, but partly uses external trucking companies for shipping its cargo. This allows the company to schedule movements in one direction without scheduling a return trip, leaving the burden of finding a lucrative return trip to the trucking company. For the proposed optimization model, this would result in a negative number of vehicles available at a hub. For this reason, and to make results more comparable, we use a relaxed version of the mathematical model for transshipment for the simulations, where we allow negative values in the decision matrix $A_{t,h}$.

Capacity of Vehicles. Considering so-called bulky freight, defining the maximal capacity of vehicles solely by weight is incomplete. However, data about the dimensions of shipments is not consistently available in the data. We therefore infer the capacity of vehicles from the datasets, based on shipment weight. In particular, the maximum vehicle capacity is inferred by observing the transported order weight per movement. The dataset shows a noticeable drop of production orders for freight that is over 9500 kg. With input from domain experts, we add a 10% safety margin, resulting in a weight-based vehicle capacity of 8550 kg. Note that the average transported weight recorded in the dataset was approximately 3500 kg. We apply this capacity constraint to all vehicles equally, as the optimization model considers a homogeneous fleet.

Simulation Environment

All framework components were implemented in Java 8 and Kotlin 1.0 and were executed in a Windows 10 Pro x64 environment (CPU: Intel Core i5-3570K @ 4.10GHz, Main Memory: 16GB). The CPLEX Solver for solving instances of the two mathematical optimization problems for computing a dynamic schedule was hosted on IBM DOCloud[1].

[1] `https://www.ibm.com/us-en/marketplace/decision-optimization-cloud`

Our usage plan offered a shared VM pool with 6 CPU-cores and 28GB of main memory and had an execution time limit of 60 minutes.

Usage Limits of DOCloud and Model Complexity. In preparatory experiments, we observed that the DOCloud solver never returned the optimal result for instances of the dynamic transshipment problem, which calculates freight streams and determines the number of vehicles per connection and time unit. For larger problem instances, such as the ones we used during the simulations, the solver did not return any result within the one hour time limit. For this reason, some measures for reducing the problem complexity were necessary. Since our dynamic transshipment model considers arbitrary predetermined routes (as a method for including previous routing results into current computations), we were able to assume arbitrary fractions of the fixed schedule to be predetermined in the dynamic transshipment instance, which improved the runtime of the solver. In a number of preparatory experiments and the help of logistics experts, we determined a suitable number for the fraction, assuming 65% of all movements from the dataset as predetermined for the simulations. The planning horizon, another complexity factor for scheduling, was set to 48 hours for the experiments, as we noticed an exponential correlation between the planning horizon and the runtime of problem instances.

7.1.2 Simulation Execution

During a simulation, the simulator issues transport orders according to the issue date in the dataset to a reference implementation of the framework for optimizing freight streams, introduced in Chapter 5. In fixed 24-hour cycles, the framework invokes the optimization agent to plan the schedule for the next 48 hours, based on the current network state and all incoming orders. The planning horizon of the last invocation is shortened to 24 hours, to stay within the limits of a work week. The first invocation run is executed on Sunday 12pm simulated time, using orders from the previous week. Note that typically, such calculations would be done in the previous week, such that there was enough lead time for employees to plan their shifts. However, we assume no lead-time in our simulations, since incidents are not considered, leading to deterministic simulations – i.e., a network snapshot prediction results in the exact same state than the actual state, without deviations. Starting from the first invocation, the optimization agent is invoked once every 24 hours and five times in total. In the following, we present the results of the simulations.

Results

The optimization agent was invoked five times per simulation, resulting in five problem instances for transshipment and vehicle allocation, i.e., the two optimization problems specified in the routing policy. Table 7.2 lists the results of both simulations for each individual invocation and aggregates the overall performance in the last two rows. In the table, O_w refers to the transport order weight in tons, $|M|_B$ and $|M|_D$ are the number of movements for the baseline and the dynamic routing policy solution, $|M_{ut}|$ refers

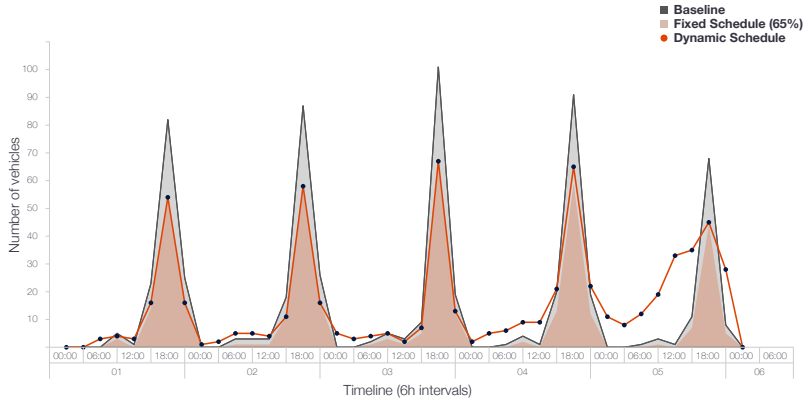
Table 7.2: Simulation Results for Q1 and Q4, Assuming no Incidents

	O_w	$\|M\|_B$	$\|M\|_D$	M_{ut}	$dist_B$	$dist_D$	gap
Inv1	633	136	96	73.2%	34239	23318	3.08%
	640	125	87	81.7%	31890	19329	2.55%
Inv2	525	140	98	59.6%	36240	27513	5.67%
	458	113	84	60.6%	29170	19003	6.17%
Inv3	456	139	107	47.3%	34788	32680	12.91%
	473	128	108	48.6%	31982	21585	14.24%
Inv4	390	136	122	35.5%	34464	31267	22.77%
	348	117	118	32.8%	29310	27156	6.21%
Inv5	180	92	186	9.7%	24244	29546	38.15%
	134	66	146	10.2%	14446	24794	29.08%
Overall	2183	643	630	38.5%	163975	144324	16,51%
	2053	549	543	42.1%	136798	111867	11,65%

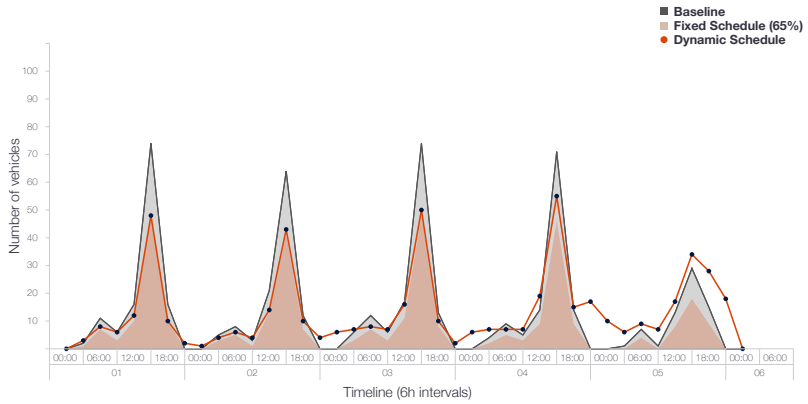
Format: B ... Baseline Schedule, D ... Dynamic Schedule

to the vehicle utilization. $dist_B$ and $dist_D$ further denote the total amount of driven kilometers for both schedules and *gap* describes the solution quality of the solution found by the CPLEX solver for the dynamic transshipment problem. It is the relative distance to the computed lower bound of the given problem instance. Our routing policy, in combination with 65% of predetermined movements from the fixed schedule showed significant improvements when compared to the fixed schedule from the data. Overall, we were able to reduce the number of movements from 643 to 630 and from 550 to 543, marking an improvement of 2% for Set Q1 and 2.3% for Set Q4. This improved the utilization of vehicles by 2.5 and 3.1 percentage points respectively. More importantly, the dynamic schedule resulted in a drastic reduction of driven kilometers by 11.9% and 19.3%, respectively.

In Figure 7.3a and Figure 7.3b, we demonstrate the difference between the static and the dynamic schedule with 65% predetermined movements. The plots represent the amount of movements in three-hour intervals during both simulations and visualize temporal distribution characteristics. The dynamic schedule operates less periodic and distributes the movements more equally on the timeline. The graphic also shows that the majority of movements are scheduled for Friday in both simulations. In principal, we have two explanations for this observation. First, the last planning horizon for I5 was shorter (24 instead of 48 hours), leaving fewer possibilities to place movements. Secondly, the amount of predetermined movements on Friday was smaller compared to other weekdays – e.g. 34% and 49% less than on Wednesday – which left more possibilities for the policy to set different times of the day, effectively spreading out the movements. On average, we recorded a shipment turnover rate of 1.62. Only 24.3% of all shipments required two or more turnovers, 12.8% three or more.



(a) Resulting Dynamic Schedule for Set Q1



(b) Resulting Dynamic Schedule for Set Q4

Figure 7.3: Comparison of Baseline Schedule and Dynamic Schedule with 65% Predetermined Routes

The maximum number of recorded turnovers was six (1% of all shipments). For shipments that required turnovers, the average duration from first loading to last offloading was 14.2 hours.

7.1.3 Solution Characteristics

Some of the solution characteristics require further explanations. For example, the utilization on Monday is much higher than the utilization at the end of the week. The decline in utilization can be explained by the balance of available vehicles per hub. We initialize the network with a certain number of vehicles per hub. During simulations, vehicles start to concentrate at consumer hubs and return trips to producer hubs become less lucrative. However, this phenomenon does not explain the particularly high solution gaps of *Inv5*. Note that the gap always was recorded for *Dynamic transshipment*, while *Vehicle Allocation* always found the optimal result.

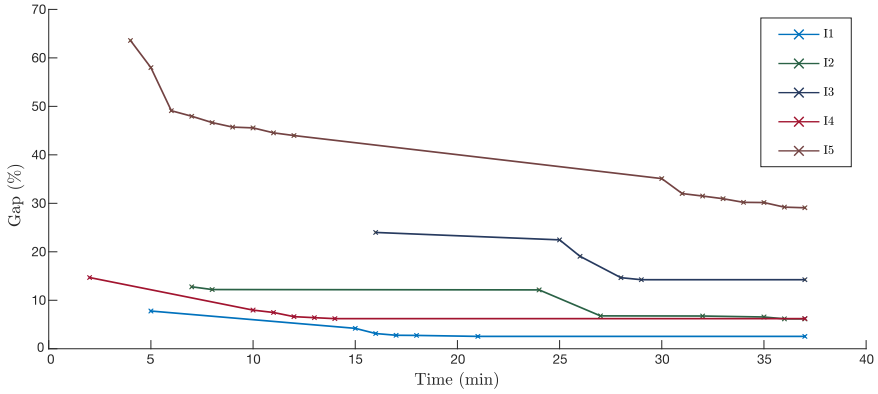


Figure 7.4: Intermediate Solution Gap for Dynamic Transshipment (*Inv1-5*)

Non-optimal solutions

Our optimization model for dynamic transshipment has two separate tasks. It finds routes for shipments through the network of hubs and determines the required number of vehicles accordingly. Therefore, for the 65% of movements that are predetermined, it only involves assigning shipments to the movements. One way of determining the solution quality is by looking at the utilization of predetermined versus newly allocated movements. For the given solutions, we found that the average utilization of predetermined movements was 23% better than newly created movements. This is intuitive, since assigning shipments to predetermined movements comes at no additional cost. Since the results from Table 7.2 indicate a growing solution gap for consecutive invocations of *Inv1-5*, we reconstructed the gaps from intermediate solutions found by the solver and reconstructed the solution convergence patterns. The results can be seen in Figure 7.4. The graphic depicts gaps from intermediate solutions for all invocations *Inv1-5*. It shows that for all cases, no better solution was found after 35 minutes and that *Inv5* started with a drastically worse initial solution.

Even though both *Inv5* problem instances were considerably smaller than the other instances in the number of transport orders and movements, paradoxically, the solution gap was higher and vehicle utilization lower. A brief investigation identifies a possible link between the reduced planning horizon of 24 hours and the solution quality. When running the same problem instances *Inv5* with a regular planning horizon of 48 hours, solution gaps improved very drastically by 35.8 and 27.1 percentage points, making the gaps comparable to *Inv1*. Although we did not further investigate this topic, one theory is plausible: Shipments that require transportation over multiple hubs introduce temporal order constraints on some movements. The theory argues that it is easier to find assignments that fulfill these constraints in an extended period of time rather than needing to "squeeze" a valid combination of movements within a 24 hour planning horizon. However, showing that this property actually holds would require more systematic evaluations.

7.1.4 Section Summary

In this section of the evaluation, we performed two simulations based on two separate transportation data sets, simulating two weeks of transportation activity. In both simulations, our dynamic vehicle routing policy was able to improve the utilization of vehicles and reduce the number of required vehicles, as well as total kilometers driven. With the large gaps in our solutions, we pointed out a weakness of our approach and discussed possible solutions. In the next section, we evaluate the performance of our subsystem for incident handling.

7.2 Performance of Incident Handling

We use the two dynamic schedules computed in the previous experiments and simulate incidents to test the subsystem for incident handling. For each incident class, namely single-vehicle, single-connection and multi-connection incidents, we simulate 10,000 instances with different delay times and measure implications to the schedule. We classify the result of each individual incident as *No Action*, *Postponement* or *Rerouting*, depending on the action that was taken by the subsystem. We assume that changes to the schedule are undesirable, which is why *No Action* is the most desirable and *Rerouting* the least desirable action.

7.2.1 Assumptions

Due to the lack of detailed data regarding traffic delays after incidents on Austrian roads, we base our assumptions on delay models from the literature, as well as input from domain experts. For example, Kwon et al. [KMV06] study the delays after incidents such as special events, lane closures, weather, and congestions on urban freeways. Their data is based on measurements from an interstate highway in a densely populated area and serves as an approximation for delays on the Austrian autobahn, which is the main type of road for LTL inter-hub transportation. In their study, the authors find that for the measured road section, incidents (i.e., accidents from other traffic participants), precipitation and congestion account for 13.3%, 1.6%, 47.4% of the total delays respectively. Other factors for delays include special events and ramp metering which our model would consider as multi-connection and single-connection delays. Note that for events known in advance, the scheduler would assume longer travel times between hubs and consider it in the dynamic schedule. In the following, we elaborate on the different types of delays we consider in our system and disclose experiment assumptions.

Single-Vehicle Delay. Want et al. [WCB05] study the duration of vehicle breakdowns on roads in the United Kingdom. Based on 1080 incident records, the authors determine a mean breakdown delay of 50 minutes for heavy goods vehicles (arithmetic mean). In addition, none of their recorded incidents lasted longer than 120 minutes. In our simulations, we approximate these values and consider two experiments, testing incidents with a duration of 0-1 hours and 1-2 hours, respectively. For better comparability with

the other two delay types, we perform three additional experiments with 2-3 hours, 3-4 hours and 4-5 hours of delay time. The selection of vehicles for which we simulate a breakdown is uniformly random and does not consider maintenance cycles or vehicle age as possible factors.

Single-Connection Delay. In our experiments, we model traffic congestion delays depending on the time of the day with peek traffic from 5am to 10am in the morning and from 3pm to 8pm in the afternoon [KMV06]. A different study [Bov98] on waiting times on roads suggests that for Austrian roads in particular, in 95.5% of all cases the delay does not exceed one hour, while 1.5% of all cases last longer than three hours. In our simulations, we perform five different experiments, testing connection delays between 0-5 hours in one-hour increments. Preparatory experiments showed that a uniformly random selection of routes had almost no effect on the schedule, since in total, there exist 121 routes. This means that on average, a route will be traveled only six times during one week. Therefore, we filtered for the 20 most heavily used routes. Note that we model connections unidirectionally, i.e., travel times between two hubs can differ, depending on the travel direction.

Multi-Connection Delay. Goodwin et al. study the impact of bad weather conditions on traffic delay [Goo02]. We base the connection delays in our experiments on their findings. In particular, we select two different speed reduction factors for rain and for snow, for which the authors found the driving speed to be reduced by 10% and 36% respectively. In the experiments, we increase the travel time for connections according to these factors. We model bad weather conditions by setting delay times for all connections that lead to or originate from a randomly selected hub.

7.2.2 Simulations

For each experiment and time unit, we simulate 10,000 incidents and record the outcome based on what action was taken. The schedules are taken from the two simulations, calculated by our routing policy.

Results

Table 7.3 summarizes the results from single vehicle and single connection incidents, i.e., all scenarios where the delay is not relative to the travel time of connections. The first column includes SV (single-vehicle) and SC (single-connection) to refer to the incident type and an interval that describes the simulated delay in minutes. The results show that the duration of the delay has a significant impact on the outcome. Note that according to the sources cited above, all real-world records of single vehicle incidents would lie within SV [0-60] and SV [60-120] and 95.5% of single connection incidents would lie within SC [0-60]. The results indicate that despite the potential effect on more vehicles from single-connection rather than single-vehicle incidents, delays from congestion have less influence on the schedule than vehicle breakdowns. This is due to the fact that most

Table 7.3: Results from Handling 10,000 Single-Vehicle and Single-Connection Incidents

Scenario	No Action	Postponement	Rerouting
SV [0-60]	7811 (78.11%)	1969 (19.69%)	220 (2.20%)
SV [60-120]	4819 (48.19%)	4805 (48.05%)	376 (3.76%)
SV [120-180]	3654 (36.54%)	5815 (58.15%)	531 (5.31%)
SV [180-240]	2941 (29.41%)	6411 (64.11%)	648 (6.48%)
SV [240-300]	2567 (25.67%)	6519 (65.19%)	914 (9.14%)
SC [0-60]	9911 (99.11%)	89 (00.89%)	0 (0.00%)
SC [60-120]	9580 (94.31%)	523 (05.14%)	54 (0.53%)
SC [120-180]	8642 (82.2 %)	1442 (13.71%)	429 (4.08%)
SC [180-240]	7475 (66.42%)	2943 (26.15%)	835 (7.42%)
SC [240-300]	6292 (49.99%)	5304 (42.14%)	990 (7.86%)

Table 7.4: Results from Handling 10,000 Multi-Connection Incidents

Scenario	No Action	Postponement	Rerouting
MC 10% [0-60]	8868 (77.08%)	1477 (12.83%)	1159 (10.07%)
MC 10% [60-120]	8112 (49.73%)	4628 (28.37%)	3571 (21.89%)
MC 10% [120-180]	8484 (38.77%)	7502 (34.28%)	5895 (26.94%)
MC 10% [180-240]	9257 (32.63%)	10838 (38.20%)	8273 (29.16%)
MC 10% [240-300]	10343 (29.03%)	14506 (40.72%)	10768 (30.23%)
MC 36% [0-60]	8785 (75.77%)	1174 (10.12%)	1635 (14.10%)
MC 36% [60-120]	7821 (47.65%)	3526 (21.48%)	5063 (30.85%)
MC 36% [120-180]	7937 (35.45%)	5979 (26.70%)	8473 (37.84%)
MC 36% [180-240]	8682 (30.42%)	8312 (29.12%)	11541 (40.44%)
MC 36% [240-300]	9403 (26.51%)	10718 (30.22%)	15342 (43.26%)

movements are scheduled outside from rush hours. On the other hand, the probability of a broken down truck increases with the number of trucks on the road and is highest during the five recorded peeks in the two schedules. For a delay of up to 60 minutes, 2.2% of all single-vehicle incidents required rerouting, while only 0.53% of all single-connection incidents on the 20 most active routes required rerouting. This number increased to 3.76% and 4.08% for delays between 60 and 120 minutes.

Table 7.4 depicts the results from multi-connection incidents due to precipitation. We simulate ten different scenarios with durations between 0 to 300 minutes – i.e. the duration of the weather condition – and travel time delays of 10% (simulating rainy weather) and 36% (simulating snowy weather) for the selected connections during that time period. The results show that all of our simulated multi-connection delays affecting one randomly selected hub had substantially larger impacts on the schedule than other types of delay. Even for short periods of precipitation, 10.07% (rain) and 14.10% (snow) of all incidents required rerouting. For snowfall scenarios that lasted longer than two

hours, rerouting was necessary in the majority of incidents (37.84% - 43.26%). Averaged over all five scenarios, an increase of travel time by 26 percentage points (from 10% to 36%) led to an increase of 28.92 percentage points of cases that required rerouting. Overall the findings show, that multi connection incidents have larger implications on the schedule than other types of delays. However, to put things in perspective, during the 10,000 simulated incidents, 11,504 and 11,595 movements were affected, i.e. 1.15 and 1.16 vehicles per incident. In the simulated worst-case scenario MC 36% [240-300], on average 3.5 vehicles were affected per incident.

7.2.3 Section Summary

In this section, we evaluated our subsystem for incident handling by classifying outcomes according to their impact on the schedule. We distinguished three outcomes and found that the majority of single-vehicle and single-connection incidents could be averted without any further action or by postponing subsequent movements. For multi-connection delays due to bad weather conditions, we noticed an increase in incidents that required rerouting. However, even for the worst case scenario, on average only 3.5 vehicles were affected by an incident.

7.3 Weaknesses and Open Issues

During the evaluation, we identified a number of shortcomings, most of which concern the design and complexity of our routing policy and consequences thereof.

7.3.1 Solution Quality

The large gaps in the quality of solutions are an issue that needs to be addressed for computing schedules. While our solutions improved the real-world data baseline, the quality of our solutions in terms of the gap to the theoretical optimum stayed behind comparable results from the literature (Noham and Tzur: 0.08% with 10 nodes [NT14], Rais et al.: 3.33 % with 14 nodes [RAC14], see Chapter 3). We explain this performance difference with the choice of CPLEX – i.e., a general problem solver – instead of a dedicated algorithm, optimized for one variant of the transshipment problem. Such an implementation would be limited in terms of requirement changes and additional constraints. It is important to note that, so far, there exist no efficient algorithm for solving the variant of the dynamic transshipment problem we defined in Chapter 6.

7.3.2 Scalability

The transshipment problem has been shown to be NP-hard [HT01], which poses a large scalability issue. While we performed no systematic experiments, a number of observations during preparatory experiments give rise to the assumption that there exist exponential runtime correlations for some of the problem dimensions. For example, we measured an exponential growth of runtime behavior for small instances, when increasing

the time dimension T; e.g., when increasing the planning horizon from ten to eleven hours, the runtime roughly doubled. The same was true for increasing the number of hubs $|H|$ in our model. This problem could be mitigated by not allowing all possible combinations of connections between hubs, i.e., by omitting direct links between hubs that are never used.

One possible way to combine the benefits of both CPLEX and domain-specific algorithms would be the application of single-solution-based meta-heuristics [Tal09] based on initial solutions from the CPLEX solver. In this scenario, the CPLEX solver would serve as a construction heuristic, i.e., a mechanism for finding a "good" initial solution, which would then be improved by a problem-specific neighborhood search algorithm. This heuristic could improve on systematic weaknesses of CPLEX solutions and improve results.

7.4 Evaluation Summary

In this chapter, both our approach for dynamic vehicle routing and our subsystem for incident handling were evaluated. By using real-world data from DB Schenker, we evaluated our routing policy, consisting of two mathematical optimization models. The results show that by replacing a fixed schedule with a dynamic schedule with 65% predetermined routes, we were able to improve the utilization of vehicles by 2.5% and 3.1%, reducing the number of movements by 2% and 2.3% and more drastically reducing the number of total kilometers driven by 11.9% and 19.3%. During the simulations, we were able to identify shortcomings, regarding our mathematical optimization model in terms of its complexity and according solution qualities resulting from invocations. For example, the overall recorded gap to optimal solutions of dynamic transshipment instances were 16.51% and 11.65%. Although the results of dynamic vehicle routing improved the performance metrics in comparison to the baseline, even more performance gains would be possible by further closing the gap to optimal solutions. For example, we found that narrowing the planning horizon of *Inv5* to 24 hours had a big impact on the gap, compared to a 48 hour calculation.

In a second evaluation, we tested the performance of our incident handling, based on the schedules from the first experiments. This evaluation brought insights not only into the resolving mechanism of incidents, but at the same time revealed stability characteristics against incidents regarding the schedule incidents, since the outcome of incident handling not only depends on the mechanism, but also on the underlaying schedule.

CHAPTER 8

Conclusions

In the last chapter of this thesis, we summarize our approach and our findings, present our vision for future work and finally conclude with a few closing remarks.

8.1 Summary

In this thesis, we presented a technical solution for improving the efficiency of LTL logistics by automating some aspects of operational decision making using rich, real-time information. First, we identified a posing problem in road logistics. Scarce information sharing between systems and companies and inflexible, long-term planning leads to inefficient schedules, a low utilization of vehicles and a large fraction of empty journeys. To solve this problem, we identified requirements and refined our assumptions by conducting a survey on related work, as well as state of the art developments and future trends in logistics, e.g., real-time tracking, autonomous and anticipatory logistics, and autonomous driving. Considering architectural guidelines for real-time decision support and autonomous decision systems, we designed a framework for collecting heterogeneous data from IoT devices in the context of logistics networks. The framework uses a number of sensors and services to maintain an information-rich digital image of the environment as a basis for routing and scheduling decisions. In a subsequent step, we used this information from the framework for optimizing freight streams by applying standard OR methods to the problem. In particular, we introduced a methodology for computing dynamic vehicle schedules on demand, based on incoming transport orders and the current state of the network in a level of detail that has previously not been achieved and which makes the approach suitable for autonomous decision making in day-to-day operations. To that end, we formally defined two mathematical optimization problems for a variant of the transshipment problem and vehicle routing which – executed sequentially – build a full vehicle schedule based on dynamic demand. In detail, we defined *Dynamic Transshipment with Time Constraints in a Finite Planning Horizon*, which optimizes freight streams

based on incoming transport orders and existing vehicle movements, considering various constraints on delivery times, capacities of vehicles, loading terminals and hubs. Based on these solutions, which represent a certain demand of vehicles per connection and time unit, we defined a second optimization problem called *Vehicle Assignment* that takes a list of available vehicles and assigns them to the demanded movements, effectively forming a full vehicle schedule. In addition to this short-term planning mechanism, we introduced a mechanism for autonomously detecting and resolving unexpected incidents such as vehicle breakdowns or traffic congestions that cause delays and cannot be foreseen by the scheduler. For evaluating both the framework, including the subsystem for incident handling, and the optimization models, we developed a simulation application, that simulates relevant aspects of the environment including parcel scans, loading terminal sensors, vehicle telemetry, weather information, and traffic information. Based on detailed LTL transportation data from an Austrian logistics company, we evaluated our approach for freight stream optimization. The results of our evaluation suggest that the dynamic schedule – initialized with 65% of the original schedule for faster computation – improved the static schedule employed by the company in all measured performance metrics. In two separate experiments, we were able to improve the utilization of vehicles by 2.5 and 3.1 percentage points, reduce the number of movements by 2% and 2.3% and more drastically reduce the number of total kilometers driven by 11.9% and 19.3%. In a second set of experiments, we tested the resulting schedules with unforeseen incidents. We simulated vehicle breakdowns, traffic delays and bad weather conditions and found that all tested scenarios impacted the computed schedule only in minor ways. Most of the time, less than two vehicles were affected by a delay and the worst case affected 3.5 vehicles on average, indicating reasonable measures for incident handling, as well as a robust schedule. Finally, with large solution gaps and scalability issues, we identified weaknesses of our approach and discussed potential solutions, which we discuss in the following section.

8.2 Future Work

To improve our contributions and to address some of its weaknesses, we outline a list of directions for further research. In addition to the shortcomings we discussed in Chapter 7, we identified several points of improvement:

- **Single-Solution-Based Meta-Heuristics**: By combining the benefits of CPLEX as a generic problem solver for declarative problem definitions with a single-solution-based meta-heuristic – e.g. local neighborhood search or similar – the large solution gaps could be mitigated. This requires a deeper analysis of solution characteristics, such as systematic weaknesses from CPLEX solutions and the development of an algorithm that improves on these weaknesses to improve solution quality.

- **Dimensional Data**: We tested our framework with weight-based capacity constraints. The integration of dimensional data, both in the test data, as well as in the optimization models could further improve the quality and confidence in our solution and provide a more detailed description about the shipped cargo.
- **Homogeneous fleet**: While the property holds for the tested transportation network, in general, logistics companies operate multiple vehicle types with different capacities and other attributes. By extending the optimization model to support multiple classes of vehicles, this problem could be addressed and the utilization of resources could further be improved by using smaller vehicles for smaller transportations.
- **Stochastic Future State Prediction**: In general, our optimization agent does not calculate vehicle schedules directly based on the current state of the network, but derives a future state from the network state to give operators some lead time with the dynamic schedule. Currently, state prediction mechanisms for calculating schedules are based on a deterministic mechanism that assumes no delays or similar incidents that change the state in unexpected ways. By using a stochastic network prediction, one could account for such incidents when predicting the future state.

8.3 Closing Remarks

For thousands of years, it was sufficient for merchants to rely on experience and intuition for transporting their goods. This has changed. In 2016, the world trade volume was 15,000 Billion USD [wto17] and in Europe alone, goods in the order of 2,200 Billion tonne-kilometers were transported, 75% of which on roads [Eur16]. In this thesis, we have demonstrated in a small example, how a modern toolset of different technologies, from IoT sensors, event-based application architectures, knowledge extraction from continuous data streams to methods for declarative problem solving can help us to make better decisions, leading to more efficient and sustainable transportation.

Acronyms

CEP Complex Event Processing. 5, 6, 11, 19, 20, 33, 36

DBMS Database Management System. 12

DSMS Data Stream Management System. 12

DSS Decision Support System. 6, 13, 14, 19, 20, 23, 25

ELR Empty Load Ratio. 19, 20

EPC Electronic Product Code. 2, 3, 11, 17, 18

FTL Full truckload. 7

FZI Forschungszentrum Informatik. 17

IaaS Infrastructure as a Service. 14

ICT Information and Communications Technology. 4, 9

IoT Internet of Things. 1, 3–7, 9–11, 14, 18, 20, 25, 28, 30, 73, 75

LTL Less than truckload. 7, 8, 23, 60, 67, 73, 74

OPL Optimization Programming Language. 40

OR Operations Research. 6, 9, 15, 16, 19, 73

PaaS Platform as a Service. 14

RFID Radio Frequency Identification. 2, 5, 10, 11, 17–19

SaaS Software as a Service. 14

TSP Traveling Salesman Problem. 20

URI Uniform Resource Identifier. 11

Bibliography

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[Bov98] Salomon Ilan Bovy, Piet H L. Congestion in europe: Measurements, spatial patterns, policies. *Transportation Planning and Traffic Engineering Report*, 1998.

[BSG08] Nabil Belgasmi, Lamjed Ben Saïd, and Khaled Ghédira. Genetic optimization of the multi-location transshipment problem with limited storage capacity. In *ECAI*, pages 563–567, 2008.

[CBM+13] Cristina Cabanillas, Anne Baumgrass, Jan Mendling, Patricia Rogetzer, and Bruno Bellovoda. Towards the enhancement of business process monitoring for complex logistics chains. In *International Conference on Business Process Management*, pages 305–317. Springer, 2013.

[CDCMB14] Cristina Cabanillas, Claudio Di Ciccio, Jan Mendling, and Anne Baumgrass. Predictive task monitoring for business processes. In *International Conference on Business Process Management*, pages 424–432. Springer, 2014.

[CGP09] Teodor Gabriel Crainic, Michel Gendreau, and Jean-Yves Potvin. Intelligent freight-transportation systems: Assessment and the contribution of operations research. *Transportation Research Part C: Emerging Technologies*, 17(6):541–557, 2009.

[Chr16] Martin Christopher. *Logistics & supply chain management*. Pearson Higher Ed, 2016.

[CM12] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012.

[Dav15] R Davies. Industry 4.0: Digitalisation for productivity and growth. *European Parliamentary Research Service, Briefing*, 2015.

[DDKS15] Wolfgang Domschke, Andreas Drexl, Robert Klein, and Armin Scholl. *Einführung in Operations Research.* Springer-Verlag, 2015.

[DHS07] Dursun Delen, Bill C Hardgrave, and Ramesh Sharda. Rfid for better supply-chain management through enhanced information visibility. *Production and Operations Management*, 16(5):613–624, 2007.

[DXHL14] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.

[EB09] Michael Eckert and François Bry. Complex event processing (cep). *Informatik-Spektrum*, 32(2):163–167, 2009.

[ec211] Roadmap to a single european transport area – towards a competitive and resource efficient transport system. Technical report, European Commission, 2011.

[EH08] Banu Yetkin Ekren and Sunderesh S Heragu. Simulation based optimization of multi-location transshipment problem with capacitated transportation. In *2008 Winter Simulation Conference*, pages 2632–2638. IEEE, 2008.

[EN10] Opher Etzion and Peter Niblett. *Event Processing in Action.* Manning Publications Co., 2010.

[Eur13] statistical office of the European Union Eurostat. Modal split of inland freight transport, 2013 (tkm), 2013.

[Eur14] statistical office of the European Union Eurostat. Share of empty journeys in the total journeys by type of operation, 2014 (

[Eur16] statistical office of the European Union Eurostat. Freight transport statistics, 2016.

[FFFM13] Zohar Feldman, Fabiana Fournier, Rod Franklin, and Andreas Metzger. Proactive event processing in action: a case study on the proactive management of transport processes (industry article). In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 97–106. ACM, 2013.

[Gia09] George A Giannopoulos. Towards a european its for freight transport and logistics: results of current eu funded research and prospects for the future. *European Transport Research Review*, 1(4):147–161, 2009.

[Goo02] Lynette C Goodwin. Weather impacts on arterial traffic flow. *Mitretek systems inc*, 2002.

[Goo05] R Goodman. Whatever you call it, just don't think of last-mile logistics, last. *Global Logistics & Supply Chain Strategies*, 9(12), 2005.

[Gud07] Timm Gudehus. *Dynamische Disposition: Strategien zur optimalen Auftrags-und Bestandsdisposition*. Springer-Verlag, 2007.

[Gud12] Timm Gudehus. *Logistik 2: Netzwerke, Systeme und Lieferketten*. Springer-Verlag, 2012.

[Gud13] Timm Gudehus. *Logistik: Grundlagen-Strategien-Anwendungen*. Springer-Verlag, 2013.

[GVdVV11] Roel Gevaers, Eddy Van de Voorde, and Thierry Vanelslander. Characteristics and typology of last-mile logistics from an innovation perspective in an urban context. *City Distribution and Urban Freight Transport: Multiple Perspectives, Edward Elgar Publishing*, pages 56–71, 2011.

[Hal85] Randolph W Hall. Vehicle scheduling at a transportation terminal with random delay en route. *Transportation Science*, 19(3):308–320, 1985.

[HRR00] Traci J Hess, Loren Paul Rees, and Terry R Rakes. Using autonomous software agents to create next generation of decision support systems. *Decision Sciences*, 31(1):1, 2000.

[HS14] Zhi-Hua Hu and Zhao-Han Sheng. A decision support system for public logistics information service management and optimization. *Decision Support Systems*, 59:219–229, 2014.

[HT00] Bruce Hoppe and Éva Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, 2000.

[HT01] Yale T Herer and Michal Tzur. The dynamic transshipment problem. *Naval Research Logistics (NRL)*, 48(5):386–408, 2001.

[HZLQ15] Qian Hao, Furen Zhang, Zeling Liu, and Lele Qin. Design of chemical industrial park integrated information management platform based on cloud computing and iot (the internet of things) technologies. *International Journal of Smart Home*, 9(4):35–46, 2015.

[JBWL06] Reiner Jedermann, Christian Behrens, Detmar Westphal, and Walter Lang. Applying autonomous sensor systems in logistics—combining sensor networks, rfids and software agents. *Sensors and Actuators A: Physical*, 132(1):370–375, 2006.

[JCL+08] Reiner Jedermann, Luis Javier Antúnez Congil, Martin Lorenz, Jan D Gehrke, Walter Lang, and Otthein Herzog. Dynamic decision making on embedded platforms in transport logistics–a case study. In *Dynamics in Logistics*, pages 191–198. Springer, 2008.

[Khu15] Archana Khurana. Variants of transshipment problem. *European Transport Research Review*, 7(2):1–19, 2015.

[KK16] Andreas Kliem and Odej Kao. Cooperative device cloud: A resource management framework for the internet of things. In *Connectivity Frameworks for Smart Devices*, pages 147–186. Springer, 2016.

[KMV06] Jaimyoung Kwon, Michael Mauch, and Pravin Varaiya. Components of congestion: Delay from incidents, special events, lane closures, weather, potential ramp metering gain, and excess demand. *Transportation Research Record: Journal of the Transportation Research Board*, (1959):84–91, 2006.

[Kra08] R van Kranenburg. The internet of things: A critique of ambient technology and the all-seeing network of rfid. *Institute of Network Cultures*, 2008.

[Kü13] Markus Kückelhaus. Low-cost sensor technology - a dhl perspective on implications and use cases for the logistics industry. Technical report, DHL Customer Solutions & Innovation in Cooperation with Fraunhofer IFF, 2013.

[Kü14] Markus Kückelhaus. Logistics trend radar 2014. Technical report, DHL Customer Solutions & Innovation, 2014.

[Kü16] Markus Kückelhaus. Logistics trend radar 2016. Technical report, DHL Customer Solutions & Innovation, 2016.

[LAP09] Rim Larbi, Gulgun Alpan, and Bernard Penz. Scheduling transshipment operations in a multiple inbound and outbound door crossdock. In *Computers & Industrial Engineering, 2009. CIE 2009. International Conference on*, pages 227–232. IEEE, 2009.

[LSD+15] Wolfgang Lueghammer, Wolfgang Schwarzbauer, Maria Dieplinger, Sebastian Kummer, Vogelauer Christian, Reinhard MOser, and Can Tihanyi. Industrie 4.0 und ihre auswirkungen auf die transportwirtschaft und logistik. Technical report, Bundesministerium für Verkehr, Innovation und Technologie, 2015.

[LSSW11] Guoqiong Liao, William Wei Song, Lei Shu, and Changxuan Wan. Using real-time event stream framework to develop rfid-based retailer supermarket systems. In *Information Systems Development*, pages 429–440. Springer, 2011.

[LW01] Hau L Lee and Seungjin Whang. Winning the last mile of e-commerce. *MIT Sloan Management Review*, 42(4):54, 2001.

[MBC15] James Macaulay, Lauren Buckalew, and Gina Chung. Internet of things in logistics - a collaborative report by dhl and cisco on implications and use

cases for the logistics industry. Technical report, DHL Customer Solutions & Innovation in Cooperation with Cisco, 2015.

[MD16] Ahmed Musa and Al-Amin Abba Dabo. A review of rfid in supply chain management: 2000–2015. *Global Journal of Flexible Systems Management*, pages 1–40, 2016.

[Mei11] Stephan Meisel. *Anticipatory optimization for dynamic decision making*, volume 51. Springer Science & Business Media, 2011.

[MGYA14] Ahmed Musa, Angappa Gunasekaran, Yahaya Yusuf, and Abdelrahman Abdelazim. Embedded devices for supply chain applications: Towards hardware integration of disparate technologies. *Expert Systems with Applications*, 41(1):137–155, 2014.

[MKRW10] Lothar März, Wilfried Krug, Oliver Rose, and Gerald Weigert. *Simulation und Optimierung in Produktion und Logistik: Praxisorientierter Leitfaden mit Fallbeispielen*. Springer-Verlag, 2010.

[MTZ16] Raef Mousheimish, Yehia Taher, and Karine Zeitouni. Automatic learning of predictive rules for complex event processing: doctoral symposium. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pages 414–417. ACM, 2016.

[NLW+12] EWT Ngai, TKP Leung, YH Wong, MCM Lee, PYF Chai, and YS Choi. Design and development of a context-aware decision support system for real-time accident handling in logistics. *Decision Support Systems*, 52(4):816–827, 2012.

[NOV15] Thomas Nebel, Jörg Ohnemus, and Steffen Viete. Industrie 4.0: Digitale (r)evolution der wirtschaft. Technical report, Zentrum für Europäische Wirtschaftsforschung GmbH, 2015.

[NT14] Reut Noham and Michal Tzur. The single and multi-item transshipment problem with fixed transshipment costs. *Naval Research Logistics (NRL)*, 61(8):637–664, 2014.

[Nwa96] Hyacinth S Nwana. Software agents: An overview. *The knowledge engineering review*, 11(03):205–244, 1996.

[PGM12] Victor Pillac, Christelle Guéret, and Andrés L Medaglia. An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*, 54(1):414–423, 2012.

[Pin15] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2015.

[RAC14] Abdur Rais, F Alvelos, and Maria Sameiro Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530–539, 2014.

[RSN16] Mikko Rinne, Monika Solanki, and Esko Nuutila. Rfid-based logistics monitoring with semantics-driven event processing. In *Proceedings of the 10th ACM international conference on distributed and event-based systems*, pages 238–245. ACM, 2016.

[SAB07] Edmund W Schuster, Stuart J Allen, and David L Brock. *Global RFID: the value of the EPCglobal network for supply chain management*. Springer Science & Business Media, 2007.

[SADP10] Aysegul Sarac, Nabil Absi, and Stéphane Dauzère-Pérès. A literature review on the impact of rfid technologies on supply chain management. *International Journal of Production Economics*, 128(1):77–95, 2010.

[SB98] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Sch12] Arne Schuldt. Multiagent coordination enabling autonomous logistics. *KI-Künstliche Intelligenz*, 26(1):91–94, 2012.

[Sch16] DB Schenker. Key figures about db schenker ag austria, 2016.

[SGFW10] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelfflé. Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things, European Commision*, 2010.

[SHG+10] Arne Schuldt, Karl Hribernik, Jan D Gehrke, Klaus-Dieter Thoben, and Otthein Herzog. Cloud computing for autonomous control in logistics. In *GI Jahrestagung (1)*, pages 305–310, 2010.

[Tal09] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[VdHW08] Wiebe Van der Hoek and Michael Wooldridge. Multi-agent systems. *Foundations of Artificial Intelligence*, 3:887–928, 2008.

[VKAB+11] Rob Van Kranenburg, Erin Anzelmo, Alessandro Bassi, Dan Caprio, Sean Dodson, and Matt Ratto. The internet of things. *A critique of ambient technology and the all-seeing network of RFID, Network Notebooks*, 2, 2011.

[VLK07] John K Visich, Suhong Li, and Basheer M Khumawala. Enhancing product recovery value in closed-loop supply chains with rfid. *Journal of Managerial Issues*, pages 436–452, 2007.

[WB12] Thomas Will and Thorsten Blecker. Rfid-driven process modifications in container logistics: Soa as a solution approach. *International Journal of Logistics Research and Applications*, 15(2):71–86, 2012.

[WCB05] WenQun Wang, Haibo Chen, and MARGARET C Bell. Vehicle breakdown duration modelling. *Journal of Transportation and Statistics*, 8(1):75–84, 2005.

[WH07] Katja Windt and Michael Hülsmann. Changing paradigms in logistics—understanding the shift from conventional control to autonomous cooperation and control. In *Understanding autonomous cooperation and control in logistics*, pages 1–16. Springer, 2007.

[WN13] Michael Watson and Derek Nelson. *Managerial analytics: An applied guide to principles, methods, tools, and best practices*. Pearson Education, 2013.

[WNLY06] Nien-Chu Wu, MA Nystrom, Tyng-Ruu Lin, and Hsiao-Cheng Yu. Challenges to global rfid adoption. *Technovation*, 26(12):1317–1323, 2006.

[wto17] World trade statistical review. Technical report, World Trade Organization, 2017.

[WWB15] S Wischmann, L Wangler, and A Botthof. Industrie 4.0: Volks-und betriebswirtschaftliche faktoren für den standort deutschland. *Berlin: BMWi*, 2015.

[Yee15] Pang Mei Yee. Omni channel logistics - a dhl perspective on implications and use cases for the logistics industry. Technical report, DHL Customer Solutions & Innovation, 2015.

[ZF10] Zeyan Zhang and Miguel Andres Figliozzi. A survey of china's logistics industry and the impacts of transport delays on importers and exporters. *Transport Reviews*, 30(2):179–194, 2010.

Appendix

Algorithm .1: Procedure to Predict the Future Network State

Data: Current snapshot of the network s_0; Time of prediction t_p

Result: Predicted snapshot of the network s_{t_p}, pending movements *pending*

$s_{t_p} \leftarrow s_0$;

instructions $\leftarrow$ instructions from s_0;

instructions $\leftarrow$ *instructions*.*filter*(λi . *i.state* $\neq$ *complete*);

events, *pending* $\leftarrow \emptyset$;

foreach *i in instructions* **do**
    if *i type MOVE* **then**
        undock $\leftarrow$ UNDOCK event for i.truck, i.origin;
        undock.timeStamp $\leftarrow$ i.execution;
        /* Departures and arrivals have been omitted */
        dock $\leftarrow$ DOCK for i.truck, i.destination;
        dock.timeStamp $\leftarrow$ i.completion;
        add undock, dock to *events*;
        if *i.execution* $< t_p$ *and i.completion* $> t_p$ **then**
            add i to *pending*;
        end
    end
    if *i type DISPATCH* **then**
        exit $\leftarrow$ CARGO_EXIT for i.cargo, i.hub;
        exit.timeStamp $\leftarrow$ i.execution;
        entry $\leftarrow$ CARGO_ENTRY for i.cargo, i.movement.hub;
        entry.timeStamp $\leftarrow$ i.movement.completion;
        add exit, entry to *events*;
    end
    /* Analogous for COLLECT and DELIVER */
    sort *events* by timestamp;
    events $\leftarrow$ *events*.*filter*(λe . *e.timestamp* $< t_p$);
    foreach *e in events* **do**
        h $\leftarrow$ lookup handler for *e*;
        $s_{t_p} \leftarrow$ h.handle(s_{t_p}, *e*);
    end
end