Platoon coordination and routing

Citation for published version (APA):
Platoon coordination and routing

Ing. K.P.J. van Doremalen (0797642)
D&C 2014.050

Master's Internship Report

Coach(es): Dr. Ir. B. Besselink
           Prof. K.H. Johansson

Supervisor: Prof. Dr. H. Nijmeijer

Eindhoven University of Technology
Department of Mechanical Engineering
Dynamics & Control

Eindhoven, December, 2014
Preface

This report is the result of a Master’s Internship at the KTH Royal Institute of Technology (Kungliga Tekniska Högskolan) in Stockholm, Sweden as part of the COMPANION project. COMPANION is a three-year European research project wherein the heavy-duty vehicle platooning concept for daily transport operations is investigated. The research project is a cooperation between different companies, i.a. KTH, Scania and Volkswagen Group.

An optimization algorithm is developed at KTH, as part of the COMPANION research project, that minimizes the total fuel consumption of a group of heavy-duty vehicles, hereby taking as an input the routes of the individual trucks. The fuel consumption is minimized by adapting the speed profiles on parts of the routes of the individual vehicles while accounting for starting times and arrival deadlines, in order to obtain fuel-optimal platoons. However, the optimization algorithm assumes that a road network and routes are given, and additionally the computational time enlarges quickly for increasing road network complexity.

The goal of this internship project is to develop an algorithm which provides the routes of the individual vehicles, using a given road network, starting points and destinations for a number of vehicles. This aspect includes the simplification of the road network, enabling more efficient computations in the optimization algorithm. The project is done in order to finish the master’s internship, which is part of the educational program of mechanical engineering at the Eindhoven University of Technology.

The report is intended for:

- Kungliga Tekniska Högskolan in order to assess the master’s internship project and as a basis and guideline for ongoing research on the subject of platoon coordination and routing.
- Prof. K.H. Johansson for the supervision on the project.
- Dr. Ir. B. Besselink for the coaching and mentoring on the project.
- BSc. S. van de Hoef as a guideline for using the developed platoon routing algorithm in combination with the optimization algorithm.
- Eindhoven University of Technology in order to assess the master’s internship project.
- Prof. Dr. H. Nijmeijer for the supervision on the project.

Stockholm, November 14, 2014

K.P.J. van Doremalen
Abstract

The demand for freight transport is increasing and will continue to do so as economies grow. Hence, traffic intensity and traffic congestion are growing issues. In addition, an increase in traffic naturally corresponds to higher fossil fuel usage and thereby a higher emission of harmful greenhouse gasses. Furthermore, fuel consumption is a growing issue as the easy to obtain fossil fuels become more scarce, resulting in increasing fuel prices. A solution which has the potential to resolve the issues regarding increasing traffic intensity, congestion, fuel consumption and GHG emissions, is HDV platooning, as will be elucidated in this report.

Vehicle platooning can be described as a group of vehicles travelling together at close inter-vehicular distances and at a set speed, while acting as one unit. Much research has been conducted on the subjects of traffic flow, fuel savings while platooning, and control perspectives for platooning. In order to investigate the heavy-duty vehicle platooning concept for daily transport operations, a three year European research project called COMPANION has been started. Already different studies, related to this work, have been conducted as part of this European research project, wherein the scope of the subjects is shifting from the local to global perspective relative to the vehicle. During one of these studies, an optimization algorithm has been developed at KTH, that minimizes the total fuel consumption of a group of vehicles, using a given road network and routes of the individual trucks. The fuel consumption is minimized by adapting the speed profiles on parts of the routes of the individual vehicles while accounting for starting times and arrival deadlines, in order to obtain fuel-optimal platoons. However, the optimization algorithm assumes that a road network and routes are given, and additionally the computational time enlarges quickly for increasing road network complexity.

Subsequently, an algorithm has been developed in this work, in order to provide the optimization algorithm with a simplified model of the road network, and providing the routes for a number of vehicles. The developed algorithm uses the original road network, as well as starting and destination points of the different transport missions as input, and determines the routes of the different transport missions using a path finding algorithm. The road network has been simplified and adapted using graph theory, and an interface between the developed algorithm and the optimization algorithm has been created, in order to be able to implement the outputs of the developed algorithm immediately into the optimization algorithm. In addition, different situations are tested and discussed which the developed algorithm may encounter, in order to show the working principles of the algorithm. One particular situation has been executed in the optimization algorithm and shows that the computational time reduces significantly for a simplified model of the road network. Furthermore, this situation shows that the fuel usage can be significantly reduced, by adapting the speed profiles in order to obtain fuel optimal platoons.
# Table of contents

**Title**

Platoon coordination and routing

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Heavy-duty vehicle platooning</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Enabling technologies for platooning</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Goal and strategy</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Report outline</td>
<td>6</td>
</tr>
<tr>
<td>1.5 The COMPANION project</td>
<td>6</td>
</tr>
<tr>
<td>2 Background</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Transport architecture</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 Mission and Transport Planner Layer</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2 Vehicle and Platoon Coordinator Layer</td>
<td>8</td>
</tr>
<tr>
<td>2.1.3 Vehicle and Inter-vehicle Controller Layer</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Problem formulation</td>
<td>9</td>
</tr>
<tr>
<td>3 Graph theory</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Undirected graph</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Directed graph</td>
<td>13</td>
</tr>
<tr>
<td>3.3 Subgraphs</td>
<td>14</td>
</tr>
<tr>
<td>3.4 Path finding algorithms</td>
<td>14</td>
</tr>
<tr>
<td>3.4.1 Dijkstra's algorithm</td>
<td>14</td>
</tr>
<tr>
<td>4 Platoon routing algorithm</td>
<td>17</td>
</tr>
<tr>
<td>5 Results</td>
<td>29</td>
</tr>
<tr>
<td>5.1 Platoon possibility</td>
<td>30</td>
</tr>
<tr>
<td>5.2 No platoon possibility</td>
<td>32</td>
</tr>
<tr>
<td>5.3 Multiple platoon possibilities</td>
<td>33</td>
</tr>
<tr>
<td>6 Conclusions and recommendations</td>
<td>35</td>
</tr>
<tr>
<td>Bibliography</td>
<td>39</td>
</tr>
<tr>
<td>Appendices</td>
<td>40</td>
</tr>
</tbody>
</table>
Table of contents

Title
Platoon coordination and routing

A Fuel saving potential
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Surface area/Vehicle cross section</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$A(X)$</td>
<td>Arc set of graph</td>
<td></td>
</tr>
<tr>
<td>$A^d(X)$</td>
<td>Unweighted directed adjacency matrix</td>
<td></td>
</tr>
<tr>
<td>$A^d_{tot}$</td>
<td>Multidimensional array</td>
<td></td>
</tr>
<tr>
<td>$A^u(X)$</td>
<td>Unweighted undirected adjacency matrix</td>
<td></td>
</tr>
<tr>
<td>$A^w(X)$</td>
<td>Weighted undirected adjacency matrix</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>Cost from neighbour to neighbour</td>
<td></td>
</tr>
<tr>
<td>$c_d$</td>
<td>Aerodynamic coefficient</td>
<td></td>
</tr>
<tr>
<td>$c_r$</td>
<td>Rolling resistance coefficient</td>
<td></td>
</tr>
<tr>
<td>coords</td>
<td>Coordinates matrix</td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td>Cost/edge weight</td>
<td></td>
</tr>
<tr>
<td>$D(X)$</td>
<td>Degree of graph</td>
<td></td>
</tr>
<tr>
<td>$E(X)$</td>
<td>Edge set of graph</td>
<td></td>
</tr>
<tr>
<td>$E(Y)$</td>
<td>Edge set of subgraph</td>
<td></td>
</tr>
<tr>
<td>$f(v)$</td>
<td>Fuel consumption per distance</td>
<td>$L/100 \text{ km}$</td>
</tr>
<tr>
<td>$F$</td>
<td>Original road network</td>
<td></td>
</tr>
<tr>
<td>$F_d$</td>
<td>Aerodynamic force</td>
<td>$N$</td>
</tr>
<tr>
<td>$F_r$</td>
<td>Rolling resistance force</td>
<td>$N$</td>
</tr>
<tr>
<td>$g$</td>
<td>Gravitational acceleration</td>
<td>$m/s^2$</td>
</tr>
<tr>
<td>$G$</td>
<td>Simplified model of the road network</td>
<td></td>
</tr>
<tr>
<td>$i$</td>
<td>Vertex in $V(X)$ for eliminating vertices</td>
<td></td>
</tr>
<tr>
<td>$j$</td>
<td>Neighbour of vertex $i$</td>
<td></td>
</tr>
<tr>
<td>$k$</td>
<td>Neighbour of vertex $i$</td>
<td></td>
</tr>
<tr>
<td>$K$</td>
<td>Number of transport missions</td>
<td></td>
</tr>
<tr>
<td>$m$</td>
<td>Gross Vehicle Weight</td>
<td>$kg$</td>
</tr>
<tr>
<td>$n^d$</td>
<td>Destination vertices of all transport missions</td>
<td></td>
</tr>
<tr>
<td>$n^s$</td>
<td>Starting vertices of all transport missions</td>
<td></td>
</tr>
<tr>
<td>$N$</td>
<td>Vector containing non platooning transport missions</td>
<td></td>
</tr>
<tr>
<td>$p$</td>
<td>Platooning possibility</td>
<td></td>
</tr>
<tr>
<td>$P$</td>
<td>Matrix containing platooning transport missions</td>
<td></td>
</tr>
<tr>
<td>pos</td>
<td>Coordinates matrix in format optimization engine</td>
<td></td>
</tr>
<tr>
<td>$Q$</td>
<td>Set of unvisited vertices</td>
<td></td>
</tr>
<tr>
<td>$r$</td>
<td>Vertex in $R$</td>
<td></td>
</tr>
<tr>
<td>$R$</td>
<td>Routes/paths of all transport missions</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{R}$</td>
<td>Routes of the individual trucks in format optimization engine</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Dimension</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>( t^d )</td>
<td>Destination times of transport missions</td>
<td>( h )</td>
</tr>
<tr>
<td>( t^s )</td>
<td>Starting times of transport missions</td>
<td>( h )</td>
</tr>
<tr>
<td>( u )</td>
<td>Vertex in ( Q ) having lowest cost</td>
<td>-</td>
</tr>
<tr>
<td>( v )</td>
<td>Vehicle's velocity</td>
<td>( m/s )</td>
</tr>
<tr>
<td>( v_{\text{max}} )</td>
<td>Maximum speed constraint</td>
<td>( km/h )</td>
</tr>
<tr>
<td>( v )</td>
<td>Vertex in ( V(X) )</td>
<td>-</td>
</tr>
<tr>
<td>( V(X) )</td>
<td>Vertex set of graph</td>
<td>-</td>
</tr>
<tr>
<td>( V(Y) )</td>
<td>Vertex set of subgraph</td>
<td>-</td>
</tr>
<tr>
<td>( \mathcal{V} )</td>
<td>Velocity profile as calculated by optimization engine</td>
<td>( km/h )</td>
</tr>
<tr>
<td>( X )</td>
<td>Graph</td>
<td>-</td>
</tr>
<tr>
<td>( Y )</td>
<td>Subgraph</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>Road gradient</td>
<td>( ^\circ )</td>
</tr>
<tr>
<td>( \alpha_d )</td>
<td>Constant related to aerodynamics</td>
<td>-</td>
</tr>
<tr>
<td>( \alpha_r )</td>
<td>Constant related to rolling resistance</td>
<td>-</td>
</tr>
<tr>
<td>( \eta )</td>
<td>Reduction factor aerodynamic coefficient</td>
<td>-</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Density</td>
<td>( kg/m^3 )</td>
</tr>
</tbody>
</table>
## List of Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>Adaptive Cruise Control.</td>
</tr>
<tr>
<td>CACC</td>
<td>Cooperative Adaptive Cruise Control.</td>
</tr>
<tr>
<td>CC</td>
<td>Cruise Control.</td>
</tr>
<tr>
<td>CFD</td>
<td>Computational Fluid Dynamics.</td>
</tr>
<tr>
<td>EU</td>
<td>European Union.</td>
</tr>
<tr>
<td>GHG</td>
<td>GreenHouse Gas.</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System.</td>
</tr>
<tr>
<td>GVW</td>
<td>Gross Vehicle Weight.</td>
</tr>
<tr>
<td>HDV</td>
<td>Heavy-Duty Vehicle.</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology.</td>
</tr>
<tr>
<td>ITF</td>
<td>International Transport Forum.</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transport Systems and services.</td>
</tr>
<tr>
<td>KTH</td>
<td>Kungliga Tekniska Högskolan.</td>
</tr>
<tr>
<td>LAC</td>
<td>Look-Ahead cruise Control.</td>
</tr>
<tr>
<td>MTPL</td>
<td>Mission and Transport Planner Layer.</td>
</tr>
<tr>
<td>V2I</td>
<td>Vehicle to Infrastructure.</td>
</tr>
<tr>
<td>V2V</td>
<td>Vehicle to Vehicle.</td>
</tr>
<tr>
<td>VICL</td>
<td>Vehicle and Inter-vehicle Controller Layer.</td>
</tr>
<tr>
<td>VPCL</td>
<td>Vehicle and Platoon Coordinator Layer.</td>
</tr>
</tbody>
</table>

i.a. Inter alia/among other things.  
 i.e. Id est/in other words.  
 e.g. Exempli gratia/for example.
1 Introduction

The demand for freight transport is increasing and will continue to do so as economies grow. The International Transport Forum (ITF) predicts an increase in freight transport up to 125% by 2050, compared to measured levels in 2010 [41]. In addition, traffic intensity is increasing due to the increasing demand for transportation services, making traffic congestion a growing issue. In 2010, almost 1.8 million heavy-duty vehicles (HDV) were in use in 15 countries of the European Union (EU), with a corresponding growth rate of 0.5% compared to 2009 [1].

An increase in traffic naturally corresponds to higher fossil fuel usage and thereby a higher emission of harmful greenhouse gasses (GHG). Furthermore, fuel consumption is a growing issue as the easy to obtain fossil fuels become more scarce, resulting in increasing fuel prices. The oil price is expected to increase by 60% by 2050, compared to prices in 2010 [41]. Additionally, the transport sector in the 27 countries of the European Union corresponds to 24% of the total greenhouse gas emissions and 28% of the total CO₂ emissions [23]. Therefore, the EU has set as goal to reduce the GHG emissions by 80 - 95% by 2050 with respect to the levels measured in 1990, which implies a reduction of 60% for the transport sector [24]. A solution which has the potential to resolve the issues regarding increasing traffic intensity, congestion, fuel consumption and GHG emissions, is HDV platooning.

1.1 Heavy-duty vehicle platooning

Vehicle platooning can be described as a group of vehicles travelling together at close inter-vehicular distances and at a set speed, while acting as one unit. The term platoon originates from a study on a single lane bus flow [45] and has become a standard term in the late 20th century. An example of four HDVs driving in a platoon formation is depicted in Figure 1.1.

When HDVs are driving in such a platoon formation, especially the following vehicles will experience a lowered air drag which leads to significant fuel savings and thereby a reduction in GHG emissions, see Appendix A for a more detailed elucidation. Experimental results show that on average a fuel reduction of 3.9 - 6.5% can be obtained for a heterogenous platoon of HDVs on a Swedish highway [3]. Other research and experiments demonstrated fuel saving potentials of 5 - 20% for HDV platooning on flat roads [9] [50].

Furthermore, driving is an unusual activity as far as people’s bodies are concerned, and therefore the longer the drive, the more likely the body is tensing and causing pain, which is referred to as driver tension. Moreover, potential shock waves arising in traffic congestion can be removed through automated platooning. Therefore, HDV platooning reduces traffic congestion

1 Platoon coordination and routing
and relieves driver tension, besides fuel savings and reductions in GHG emissions, without compromising safety and is expected to improve the overall traffic flow by an automated control strategy [3].

Early studies on vehicle platooning were mainly done in order to study traffic dynamics with the purpose to understand and develop traffic flow models. One of these early studies on traffic dynamics resulted in finding wave phenomena, when vehicles resumed the road speed after a traffic light turned green [43]. The attenuation of these wave phenomena, i.e. the attenuation of a disturbance in position, speed, or acceleration as it propagates along the string of vehicles, is known as string stability and was introduced by a study in relative motion [42]. Much research has been conducted during a few decades into the research field of string stability within vehicle platooning [12] [31] [33] [48] [54]. The research of vehicle platooning and string stability were mainly theoretical and new research areas started to arise in the 1990’s when technology was more mature for implementing and testing of vehicle platooning in practice, where most studies focussed on highway throughput [11] [27]. However, most recent studies were done in order to study fuel savings for platooning [4] [9] [10] [15] [49] [50] [56], wherein the focus was mainly on HDVs where the potentials are greater due to the shape and mass of such vehicles. Additionally, many recent studies have focussed on control perspectives, for example collision avoidance [5] [7]. In order to achieve and maintain a safe HDV platoon, different communication and control technologies will have to work together. Therefore, the next section will elaborate on the technologies that enable a platoon formation.

Figure 1.1: Four heavy-duty vehicles driving in platoon formation [47]

2 Platoon coordination and routing
1.2 Enabling technologies for platooning

An overview, from local to global level, of the technologies that enable HDV platooning is depicted in Figure 1.2. On local level, technologies within a small range of the vehicle are effective, such as Cruise Control (CC) and Adaptive Cruise Control (ACC). The ACC system is an extension on the CC system and has been considered as a means to enable vehicle platooning [26] [44].

Cruise Control is a system which takes over the throttle of the vehicle to maintain a steady speed as set by the driver. The throttle valve controls the power and speed of the engine by limiting the amount of intake air and is actuated automatically, instead of by pressing a pedal, when the cruise control system is engaged [29].

The Adaptive Cruise Control system is an extension on the CC system, which automatically adjusts the vehicle speed to maintain a safe distance from vehicles ahead. ACC uses either radar or laser sensors to detect the speed of and distance to the vehicle ahead. If the distance to a vehicle or object ahead diminishes, the system will send a signal to the engine or braking system in order to decelerate the vehicle and the other way around for increasing distance [29].

An extension on the ACC system is Cooperative ACC (CACC), which realizes longitudinal automated vehicle control by accounting for road information, such as road grade, and traffic events occurring further ahead in the platoon, such as traffic congestion. This is realized by wireless communication in short and wide range relative to the vehicle, i.e. by Vehicle to Vehicle (V2V) communication and Vehicle to Infrastructure (V2I) technology, respectively [3] [35] [38]. The interaction between vehicles is enabled through V2V communication and can therefore improve safety. By combining Global Positioning Systems (GPS) and V2V technology, the relative position estimates of neighbouring vehicles can be made with high accuracy. Hence, smoother control can be implemented through prediction based upon the gathered information, enabling cooperative driving and realizing automated vehicle platoons [3] [19] [46].

On global level, implementation of routing and road information are enabled through V2I technology. A command center or fleet manager can monitor the vehicles in real-time traffic through V2I, enabling the possibility to react upon road and traffic information and optimize the transport mission and thereby the vehicle’s fuel consumption. For example, fuel consumption can be decreased by adjusting the vehicle’s speed in order to form a platoon. Furthermore, an alternative path can be found through V2I, to ensure the arrival deadline of the transport mission when obstructing traffic situations are encountered [3] [8] [46].

Technologies such as V2V and V2I are part of Intelligent Transport Systems and services (ITS), where ITS denotes the integration of Information and Communication Technology (ICT) with transport infrastructure, vehicles and users [21]. Figure 1.3 represents an illustration of ITS, where ITS includes all types of communication in and between vehicles (V2V communication) along with communication between vehicles and infrastructure (V2I communication). With the aid of these communication devices, a cooperative system is formed for supporting and replacing human functions in various driving processes in order to enhance operational performance, mobility, environmental benefits, and safety [3].
Figure 1.2: Overview from local to global level of enabling technologies for HDV platooning [3]

Figure 1.3: An illustration of ITS [22]

4 Platoon coordination and routing
1.3 Goal and strategy

Different studies have been conducted related to this work, wherein the scope of the subjects is shifting from the local to global perspective relative to the vehicle. In these studies, the use of ITS was investigated in order to reduce the vehicle’s fuel consumption, for example by using preview information on road topography, which is called Look-Ahead cruise Control (LAC). An investigation in fuel savings using Cooperative Adaptive Cruise Control (CACC) has been done by Alam [3], where the focus was mainly on researching the control perspectives related to inter-vehicular control. This thesis concluded that a platoon control could be further improved and be more fuel efficient using preview information of the road topography. In addition, a study on the subject of LAC by Alam et al. concludes that it is both fuel-efficient and desirable in practice, to consider preview information of the road topography in the control strategy [6].

Furthermore, ITS can be used in order to coordinate scattered vehicles on a road network, because vehicles require a regional or global perception of the vehicles’ surroundings (V2V and V2I technology) in order to form platoons. A study on the subject of vehicle coordination by Liang concludes that fuel savings and the platooning rate, which denotes the distance platooned over the distance driven, can be increased significantly by coordinating vehicles on a road network [34].

In addition, vehicles have different origins and destinations with different starting times and arrival deadlines, i.e. vehicle platoons will have to be formed, merged and split frequently, which leads to the vehicle routing problem. Vehicle routing denotes the route of a fleet of vehicles to deliver or pick-up goods from customers at minimum costs. So in order to fully exploit the fuel saving potential, vehicle coordination and routing should be applied.

In order to compute the fuel savings for driving in a platoon formation, an optimization algorithm is developed at KTH. The optimization algorithm minimizes the total fuel consumption of a group of heavy-duty vehicles, by adapting the speed profiles on parts of the routes of the individual vehicles while accounting for starting times and arrival deadlines, in order to obtain fuel-optimal platoons. This means that the optimization algorithm assumes that a road network and routes are given, and additionally the computational time enlarges quickly for increasing road network complexity [51].

Subsequently, a platoon routing algorithm has been developed in this work, in order to provide the optimization algorithm with the routes of the individual trucks and a highly simplified road network, enabling more efficient computations in the optimization algorithm. The developed algorithm uses the original road network, as well as starting and destination points of the different transport missions as input, and determines the routes of the different transport missions using a path finding algorithm. The road network has been simplified and adapted using graph theory, and an interface between the developed algorithm and the optimization algorithm has been created, in order to be able to implement the outputs of the developed algorithm immediately into the optimization algorithm. See Section 2.2 for a more thorough explanation on the problem formulation.
1.4 Report outline

As introduced in the former sections, HDV platooning has the potential to provide significant fuel savings. In order to fully exploit the fuel savings, vehicles should be coordinated and the most fuel-optimal routes should be determined. Chapter 2 will therefore provide a transport architecture in order to decompose the overall problem of fuel-optimal goods transportation into subproblems. Furthermore, the problem formulation will be discussed in order to understand the scope of the optimization algorithm, as well as the scope of the platoon routing algorithm, and in order to understand the interface between these two algorithms. Chapter 3 will elaborate on the used graph theory, which is needed in order to fully understand the used approaches in the platoon routing algorithm. Chapter 4 will explain the working principles and approaches of the platoon routing algorithm, which is the main focus of this project. Chapter 5 will show some situations, and the corresponding results, which the platoon routing algorithm may encounter. In the last chapter of this report, Chapter 6, the conclusions and recommendations will be presented as a guideline for future research on the subject of HDV platoon coordination and routing.

1.5 The COMPANION project

A number of ITS projects have been conducted worldwide, of which one is the COMPANION project. The coordination and routing project will be part of the COMPANION project, which is a three year (2013-2016) European research project that investigates the heavy-duty vehicle platooning concept for daily transport operations. The project is a cooperation between different companies, i.a. Kungliga Tekniska Högskolan (KTH), Scania and Volkswagen Group, and has as goal to develop a real-time coordination system that dynamically creates, maintains and dissolves HDV platoons, according to a decision-making mechanism. This is achieved by taking into account both historical and real-time information about the state of the infrastructure (such as traffic congestion and weather). The consequence is that platoons will be no more composed just of vehicles with common origins and destinations, but they will be created dynamically on the road, by merging individual vehicles or sub-platoons that share subparts of their routes [13].

Different studies, which are related to the coordination and routing project, have already been conducted as part of the COMPANION project at KTH, where the scope of the subjects is shifting from the local to global perspective relative to the vehicle, see Section 1.3 and [3] [6] [34] [51].
2 Background

As introduced in the previous chapter, scattered vehicles on a road network will have to be coordinated in order to form platoons. Section 2.1 covers a transport architecture, in order to decompose the overall objective of fuel-optimal goods transportation into smaller subproblems, of which one contains the coordination and routing problem. Section 2.2 will discuss the problem formulation of the optimization algorithm, and of the platoon routing algorithm as well as the interface between these algorithms.

2.1 Transport architecture

The freight transport architecture as depicted in Figure 2.1 is proposed by Liang [34] and is inspired by the control architectures for vehicle platooning by Alam [3], Horowitz [28] and Varaiya [52]. The aim of this transport architecture is to decompose the overall objective of fuel-optimal goods transportation into hierarchical smaller subproblems, because subproblems are more easily solved than the overall objective. The proposed architecture is divided into three layers (from top to bottom): Mission and Transport Planner Layer (MTPL), Vehicle and Platoon Coordinator Layer (VPCL), and Vehicle and Inter-vehicle Controller Layer (VICL). In the following sections, the tasks and objectives of each layer will be described.

Figure 2.1: Three layer hierarchical transport architecture
2.1.1 Mission and Transport Planner Layer

A transport mission is an assignment to transport goods from one location to another within a certain time window. A transport mission is generally assigned to drivers by a fleet owner and can be registered days or months before the transport assignment. The main tasks of the Mission and Transport Planner Layer (MTPL) are distributing and assigning the transport missions over the available HDVs that are provided by one or several fleets, determining fuel-efficient routes, and coordinating transport assignments such that platoons can be formed for further fuel-efficiency. This means that MTPL handles transport planning, routing, and vehicle coordination.

The objective of transport planning is to maximize the utilization of the available vehicles as well as the cargo capacity of individual or several HDVs by grouping similar transport missions. Thereby, deadlines and physical parameters, such as weight and size, have to be taken into account, which serve as underlying constraints for optimizing the transport mission, with the objective of minimizing fuel consumption.

The objective of routing is to find the most fuel-efficient path by considering, for instance, constraints set by the road topography. Furthermore, the state of infrastructure, traffic, and environment, e.g. weather, as well as driver's resting times are taken into account in the route calculation in order to obtain reliable assignment plans. An important aspect of fuel-optimal routing is that routes will be coordinated between a number of HDVs, such that vehicles can benefit from platooning on overlapping sections of their transport missions. Consequently, the fuel-optimal routing will include the starting times and arrival deadlines of each transport mission in order to calculate whether vehicles can form a platoon, or not.

The MTPL will therefore send down information regarding time limits to the Vehicle and Platoon Coordination Layer (VPCL). The MTPL will also receive information from the VPCL, namely about the assignment status, i.e. if an assignment or part of an assignment is finished, or whenever an assignment cannot be fulfilled. With this information, the MTPL has to reiterate its plan and either reallocate the assignments or give a new time suggestion, depending on the cause of not being able to fulfill the earlier proposed plan [3] [34].

2.1.2 Vehicle and Platoon Coordinator Layer

The main task of the VPCL is to execute the transport missions with high fuel efficiency, which is mainly done by forming fuel-optimal platoons, using V2V communication and using the gathered information from the MTPL. The challenges in this layer lie in forming, maintaining and splitting platoons. In order to form a platoon, the vehicles have to adjust their speed in order to meet each other at a certain point. This is done using the given speed profile of the MTPL, which consists of average velocities over upcoming road segments in order to merge vehicles into a platoon or to meet the specified time constraints. Furthermore, maintaining a fuel-optimal platoon with a desired small inter-vehicular distance, without compromising safety, can be quite challenging due to uncertainties in vehicle parameters, sensor information and map data.

The VPCL will send down information to the Vehicle and Inter-vehicle Controller Layer (VICL) regarding reference speeds, routes and vehicles in the platoon. The VICL will report to the MTPL about the assignment status. In addition, the VPCL receives information from the VICL.
about the vehicle status and issues regarding current traffic. Vehicle status can be whether the vehicle is platooning or not, or updated estimates of vehicle parameters, such as mass [3] [34].

2.1.3 Vehicle and Inter-vehicle Controller Layer

The VICL is the vehicle itself and has therefore as main task to ensure tracking of the desired velocities and acceleration requests, which are gathered from the VPCL. In addition, fuel-efficient control strategies are executed based on on-board sensor information and the powertrain is optimized for energy efficiency (velocities, torque, gear shift strategies). Furthermore, the handling of resonances for functionality, drivability, and noise reduction, are challenges which are addressed in this layer.

If the VICL considers that the vehicle cannot maintain the speed profile given by the VPCL, due to physical constraints such as engine properties or sudden traffic accident, it will inform VPCL as a vehicle status or traffic problem, respectively, and will request for a new suggestion from VPCL. Note that the request from VICL for a new suggestion can go all the way up to MTPL if VPCL cannot find a new solution [3] [34].

From the proposed freight transport architecture, it can be concluded that the scope of the coordination and routing, as well as the scope of the fuel consumption optimization is in the Mission and Transport Planner Layer. In the next section, the problem formulation, as well as the interface between the platoon routing and optimization algorithm, will be discussed.

2.2 Problem formulation

As explained up to now, the vehicle's fuel consumption can be significantly reduced by driving in platoon formation, where vehicle coordination and routing should be applied in order to fully exploit the fuel saving potentials. As explained in the previous section, the objective of fuel-optimal routing is to find the most fuel-efficient path and corresponding velocity profile while taking into account the starting times and arrival deadlines of the different transport missions. This can be summarized as a minimization problem, which minimizes the total fuel consumption of each transport mission by optimizing over the corresponding routes \( R \) and velocity profiles \( V \), while being subject to the road network, as well as velocity and time constraints [51]:

\[
\min_{R, V} \text{fuel}(R, V) \tag{2.1}
\]

However, the fuel-optimal routing will be split into a platoon routing algorithm, and an optimization algorithm, in order to separate the routing from the fuel consumption optimization. As explained before, the platoon routing algorithm will provide the optimization algorithm with a simplified model of the road network in order to enable more efficient computations in the optimization algorithm. The road network is simplified, because the computational time enlarges quickly for increasing road network complexity. Furthermore, the platoon routing algorithm will provide the routes of the individual trucks.
The optimization algorithm uses a common simple fuel model to determine the instantaneous fuel consumption per unit distance \( f(v) \), which is a second order polynomial in the speed:

\[
f(v) = \alpha_r + \eta \alpha_d v^2, \quad \alpha_r, \alpha_d \in \mathbb{R},
\]

where \( \alpha_r \) denotes a constant related to rolling resistance and \( \alpha_d \) denotes a constant related to aerodynamics. Other influences, such as engine and brake forces, selected gear and road gradient are neglected here for convenience. When platooning as a follower, the aerodynamic coefficient is assumed to diminish by a factor \( 0 < \eta \leq 1 \). Note that if \( \eta = 1 \), the fuel consumption is determined without platooning [51]. Furthermore, the lead-vehicle will also experience a reduction in aerodynamic drag, as elucidated in Appendix A.

In order to illustrate the in- and outputs, as well as the interface between the platoon routing and optimization algorithm, a schematic view is depicted in Figure 2.2. The optimization algorithm will assume that the road network \( G \), routes \( R \) and starting times \( t_s \) and arrival deadlines \( t_d \) are given, and minimizes the total fuel consumption \( f(v) \) by optimizing over the corresponding velocity profiles \( V \).

The platoon routing algorithm will therefore provide the optimization algorithm with a simplified model of the road network \( G \) and routes \( R \) of the individual trucks, having the original road network \( F \) and starting \( n_s \) and destination \( n_d \) points as input. The platoon routing algorithm was developed using Matlab and the simplification and visualization of the road networks are made using graph theory, which leads to the introduction of this subject in the next chapter.

![Figure 2.2: Schematic view of the in- and outputs of the platoon routing and optimization algorithm](image-url)
3 Graph theory

In this chapter the preliminaries are presented, in order to fully understand the approach and working principles of the platoon routing algorithm. Terms like graph, vertex, edge, arc, adjacency matrix, degree and path are introduced throughout the chapter. Additionally, a path finding algorithm are introduced, which is used in order to find routes in a road network. In this report, road networks will be represented by graphs, which leads to the introduction of an undirected graph.

3.1 Undirected graph

An undirected graph \(X\) consists of a vertex set \(V(X)\) and an edge set \(E(X)\), where an edge is an unordered pair of distinct vertices of \(X\), such that \(E(X) \subseteq V(X) \times V(X)\). If \(\{x, y\}\) denotes an edge, such that \(\{x, y\} \in E(X)\), then \(x\) and \(y\) are adjacent vertices, i.e. \(y\) is a neighbour of \(x\). In order to visualize a graph, the vertices will be drawn as dots and the edges as lines [25], see Figure 3.1 for illustration. Note that since the graph is undirected one could travel from \(x\) to \(y\) or from \(y\) to \(x\), i.e. \(\{x, y\} = \{y, x\}\).

Figure 3.1: Example of a weighted undirected graph

One of the naturally associated matrices of a graph is the adjacency matrix. The adjacency matrix \(A^u(X)\) of an undirected graph \(X\) is a symmetric integer matrix, wherein the rows and columns denote the vertices of the graph, such that the number of nonzero entries equals the number of edges. Furthermore, the considered graphs will not contain self-loops, where a self-loop is an edge such that the ends of the edge are on the same vertex. Hence, the diagonal
entries are always zero [25]. This means that $A^u(X)$ is defined by

$$A^u_{xy} = \begin{cases} 1 & \text{if } \{x, y\} \in E(X) \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

Note that the edge weights are ignored for an undirect adjacency matrix $A^u(X)$. Considering the example of Figure 3.1, it can be seen that for instance vertex one is adjacent to vertex three and therefore vertex three is also adjacent to vertex one, resulting in a nonzero entry for $A^u_{13}(X)$ and $A^u_{31}(X)$, respectively. Continuing this approach for all vertices results in the undirected adjacency matrix

$$A^u(X) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad (3.2)$$

In order to include weights, the weighted undirected adjacency matrix $A^w(X)$ is introduced, where $A^w(X)$ is defined by

$$A^w_{xy} = \begin{cases} C_{xy} & \text{if } \{x, y\} \in E(X) \\ 0 & \text{otherwise} \end{cases}, \quad (3.3)$$

where $C_{xy}$ denotes the edge weight of the edge $\{x, y\}$. Considering again the example of Figure 3.1, it can be seen that for instance the weight from vertex one to vertex three equals three, i.e. $C_{13} = 3$, and therefore the entries $A^w_{13}(X)$ and $A^w_{31}(X)$ will equal this edge weight. This results in the matrix

$$A^w(X) = \begin{bmatrix} 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 \\ 3 & 1 & 0 & 7 & 5 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 3 & 5 & 0 & 0 \end{bmatrix}, \quad (3.4)$$

Furthermore, the degree (or valency) of a vertex $x$ is the number of neighbours of $x$. The degree of all vertices of a graph $X$ is a diagonal matrix $D(X)$, indexed by the vertex set $V(X)$ such that $D_{xx}$ is the degree of $x$ [25]. The degree of all vertices can be determined by summing up either the columns or the rows of the unweighted undirected adjacency matrix $A^u(X)$, which for the example in Figure 3.1 results in the diagonal matrix

$$D(X) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}. \quad (3.5)$$

As can be seen, for instance vertex three has indeed four neighbours, $D_{33} = 4$.

Up to now, only an undirected graph is considered, i.e. one can travel in two directions on an edge. However, only the definition of an undirected graph is not sufficient, for example when one-way streets exist in a road network. Therefore, a directed graph is introduced in order to be able to implement unidirectional edges.
### 3.2 Directed graph

A directed graph $X$ consists of a vertex set $V(X)$ and an arc set $A(X)$, where an arc denotes a directed edge, and is therefore an ordered pair of distinct vertices. If $(x, y)$ denotes an arc, such that $(x, y) \in A(X)$, then $x$ and $y$ are adjacent vertices, i.e. $y$ is a neighbour of $x$. Note that an arc is an ordered pair of distinct vertices, i.e. $(x, y) \neq (y, x)$. In a drawing of a directed graph, an arc will be indicated by an arrow $\rightarrow$, see Figure 3.2 for illustration.

![Figure 3.2: Example of a weighted directed graph](image)

Associated with a directed graph is again the adjacency matrix, where the adjacency matrix $A^d(X)$ of a directed graph $X$ is an integer matrix wherein the columns represent the starting vertices and the rows the end vertices of the arcs $\rightarrow$. This means that $A^d(X)$ is defined by

$$A^d_{yx} = \begin{cases} 1 & \text{if } (x, y) \in A(X) \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

For the example depicted in Figure 3.2 one can see that for instance vertex one points towards vertex three, resulting in a nonzero entry in $A^d_{31}(X)$. Continuing this approach for all vertices results in the adjacency matrix

$$A^d(X) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}. \quad (3.7)$$

Additionally, also a directed graph has a weighted adjacency matrix, but the weighted directed adjacency matrix will not be considered in this report.

If a transport mission exists in the example of Figure 3.2, starting at vertex one, and finishing at vertex five by traversing the vertices one, three and five, then the vertices to be traversed define the route of the transport mission, and the vertices two and four have obviously become unnecessary. An original graph can thus be simplified by obtaining a route between starting and destination vertex, and thereby computational time can be won in the optimization algorithm. Furthermore, the route can be represented as a subgraph of the supergraph $X$, which leads to the introduction of subgraphs.
3.3 Subgraphs

A subgraph of a graph $X$ is a graph $Y$ such that its vertex and edge set form a subset of the vertex and edge set of the graph $X$;

$$V(Y) \subseteq V(X), \quad E(Y) \subseteq E(X). \quad (3.8)$$

Furthermore, a subgraph $Y$ of $X$ is an induced subgraph if two vertices of $V(Y)$ are adjacent in $Y$ if and only if they are adjacent in $X$. Any induced subgraph of $X$ can therefore be obtained by deleting some of the vertices from $X$, along with any edges that contain a deleted vertex [25].

A path $R$ (or route) is defined as a sequence of distinct vertices $v_1, \ldots, v_r$, such that consecutive vertices are adjacent and that there exists an arc between each two consecutive vertices in $R$, i.e. $(v_i, v_{i+1}) \in A(X)$ for $i = 1, \ldots, r - 1$, where $v_1 = n^s$ and $v_r = n^d$. If there exists a path $R$ between any two vertices of an induced subgraph $Y$, then the subgraph $Y$ is called a connected component of $X$ [25]. In order to find paths in a graph, the platoon routing algorithm will use a path finding algorithm.

3.4 Path finding algorithms

Path finding algorithms are methods to find a (usually shortest) path in a graph between a starting and destination point, which is done by exploring adjacent vertices. Basic algorithms, such as breadth-first search and depth-first search, take all possibilities into account and are therefore time consuming algorithms. A more common path finding algorithm is Dijkstra’s algorithm, because this algorithm uses a more strategic approach by eliminating paths or vertices which are ‘unnecessary’ (longer distance) or impossible to take [14]. Dijkstra’s algorithm will be used in the platoon routing algorithm and will therefore be explained in Section 3.4.1.

3.4.1 Dijkstra’s algorithm

Dijkstra’s algorithm finds the shortest path, for each transport mission $k$, in a graph between a starting and destination vertex, by starting at an initial vertex $n^s_k$ with a set of unvisited vertices $Q$. In each step of the algorithm, the vertices of the unvisited set are examined and the vertex with the shortest distance to the current vertex will be marked as visited. The process will repeat until the destination vertex $n^d_k$ has been found [2] [14] [16] [18] [32]. The algorithm is provided in pseudocode in Algorithm 1 and will be explained on the basis of an example.

Algorithm 1 Dijkstra

1: procedure Dijkstra($A^w(X), n^s, n^d$)
2: for $k \in \{1, \ldots, K\}$ do
3: $Q \leftarrow \emptyset$, $R_k \leftarrow 0$, $C_{k,n^s_k} \leftarrow 0$
4: for each vertex $v \in V(X)$ do
5: if $v \neq n^s_k$ then
6: $C_{k,v} \leftarrow \infty$
7: $p_{k,v} \leftarrow$ undefined
8: end if

14 Platoon coordination and routing
As stated in Line 1, Dijkstra’s algorithm will use the weighted undirected adjacency matrix \( A(X) \) of the original graph, as well as the starting \( n_s \) and destination \( n_d \) vertices of all transport missions \( K \) as input. Then, the algorithm sets the weight \( C \), i.e. the cost, from the current starting vertex \( n_s \) to itself to zero, Line 3. Furthermore, the algorithm sets the distance to all other vertices to the tentative distance infinity, Lines 4-10, and adds all vertices to a set of unvisited vertices \( Q \), Line 9.

While the set of unvisited vertices is not empty, Line 12-28, the algorithm will determine the vertex \( u \) in the set of unvisited vertices \( Q \), having the lowest cost \( C_{k,u} \), Line 13. The vertex with the lowest cost will be relaxed from the unvisited set, Line 14, and all unvisited neighbours \( v \) of \( u \) will be considered and their tentative distances will be calculated, Lines 21-27. The newly calculated tentative distance \( C_{k,v} \) is compared to the current assigned cost \( C_{k,v} \), Line 22, and the smaller one is assigned, Line 23. In addition, the vertex having the lowest cost is assigned as the previous vertex and stored in the matrix \( R_k \), containing the vertices to be traversed, Line 24-25. When the vertex \( u \) equals the destination vertex \( n_d \), the process will be terminated, Lines 16-19.

Imagine that one travels from vertex one, \( n^r = 1 \), to vertex five, \( n^d = 5 \) (marked yellow), in Figure 3.3, then first the cost (depicted in red) to the starting vertex is set to zero, i.e. \( C_1 = 0 \), and the cost to all other vertices is set to the tentative distance infinity, see Figure 3.3(a). Furthermore, the previous vertex for all unvisited vertices is marked as undefined, i.e. \( p_v = ? \) (depicted in blue). Next, the algorithm examines the distance from the current (marked grey) to the neighbouring vertex in the unvisited set \( Q \), which is vertex three, and calculates the new cost, where \( C_3 = 3 \). Then the newly calculated cost is compared to the current assigned value (obviously \( C_3 = 3 < \infty \)), and the lowest cost is assigned. Additionally, the previous vertex will be set as

---

15 Platoon coordination and routing
the vertex with the lowest cost, i.e. $p_3 = 1$. Furthermore, vertex one will be deleted from the set of unvisited vertices $Q$ and the next current vertex will be vertex three, because vertex three is the vertex having the lowest cost in the unvisited set $Q$.

The algorithm examines again the cost from the current vertex to its neighbours in the unvisited set, which are vertex two and five, see Figure 3.3(b). For both, the newly calculated cost is smaller than infinity, and therefore the new costs are assigned to these vertices. In addition, the previous vertex is set to the vertex having the lowest cost, i.e. $p_2 = p_5 = 3$. As can be seen, the vertex with the new lowest cost is vertex two, since $C_2 = 4 < C_5 = 8$, and therefore vertex three is deleted from the set of unvisited vertices, and the next current vertex will be vertex two. As can be seen from Figure 3.3(c), the current assigned cost to vertex five equals eight, $C_5 = 8$, and the current assigned previous vertex is vertex three, i.e. $p_5 = 3$. However, the algorithm will determine that the total cost via vertex two to vertex five equals seven, $C_5 = 7$. Hence, the algorithm will assign the smaller cost value, and the corresponding previous vertex, i.e. $p_5 = 2$, and the shortest path, $R = [1\ 3\ 2\ 5]$, from start to destination has been found (depicted with blue arrows), see Figure 3.3(d).

Figure 3.3: Example of Dijkstra's algorithm on a weighted directed graph
4 Platoon routing algorithm

This chapter will elaborate on the approach and working principles of the developed platoon routing algorithm. One can imagine, that for example the European road network will contain many vertices and arcs, in order to denote road junctions/intersections, highway entries/exits, but also in order to create for instance curves in the road network for visualization. The optimization algorithm minimizes the instantaneous fuel consumption by optimizing the velocities of the different vehicles on each arc in its route. The computational time will therefore enlarge quickly for increasing number of arcs, and thus with increasing road network complexity. Therefore, the platoon routing algorithm will determine the routes corresponding to the starting and destination points of the different transport missions. The routes represent subgraphs of the original supergraph and form therefore already a simplification of the original graph. Additionally, the routes will be simplified in order to minimize the number of arcs in the routes. This is done by eliminating vertices and corresponding arcs, such that only the starting and destination vertices, as well as the vertices on which vehicles can form or split up a platoon, remain.

The platoon routing algorithm will first determine the shortest routes between the starting and destination vertices of the different transport missions. When the routes are determined, the algorithm will have to determine whether the routes of the different transport missions share arcs, or not, i.e. if they share parts of their routes such that they may be able to form a platoon if their time constraints allow so. The routes of the transport missions which share arcs will then be merged, in order to obtain one graph containing only the vehicles which may be able to form a platoon. The routes of the vehicles that cannot form a platoon will be kept separate. After the graph of all vehicles which may be able to form a platoon is obtained, this graph, as well as the graphs of the vehicles which are not able to form a platoon, will be simplified by eliminating vertices from the routes. Furthermore, at the end of the algorithm, the obtained matrices will be converted into the correct format and will be written into a text-file, in order to be able to implement the matrices immediately into the optimization algorithm.

The overall algorithm has been split into six subalgorithms for convenience and in order to explain the algorithm part by part. The first subalgorithm is Dijkstra’s algorithm, see Algorithm 1 in Section 3.4.1. Since each computed path represents a subgraph of the original graph, each path will have its own adjacency matrices.

In order to obtain the adjacency matrices corresponding to each transport mission, Algorithm 2 will be used. This subalgorithm determines the adjacency matrices $A^t_k(X)$, $A^w_k(X)$ and $A^d_k(X)$ corresponding to the route $R_k$, where a route is defined as a sequence of vertices, i.e. $R_k = v_1, \ldots, v_{r_k}$ with $v_1 = n^t_k$ and $v_{r_k} = n^d_k$. Here, $r_k$ denotes the number of vertices to be traversed by transport mission $k$, with $k = 1, \ldots, K$, and $K$ the number of transport missions under consideration.
Algorithm 2 Adjacency matrices per route

1: procedure ADJACENCY MATRICES PER ROUTE($A^w(X)$, $R_k$)
2: \[ A^u_k(X) \leftarrow 0, \ A^w_k(X) \leftarrow 0, \ A^d_k(X) \leftarrow 0 \]
3: for $i = 1, \ldots, r_k - 1$ do
4: \[ A^u_k, R_k, i \leftarrow 1, \ A^w_k, R_k, i \leftarrow 1 \]
5: \[ A^w_k, R_k, i \leftarrow A^w_k, R_k, i+1, \ A^w_k, R_k, i-1 \leftarrow A^w_k, R_k, i+1 \]
6: \[ A^d_k, R_k, i \leftarrow 1 \]
7: end for
8: end procedure($A^u_k(X)$, $A^d_k(X)$, $A^w_k(X)$)

As stated in Line 2, the algorithm begins with the empty adjacency matrices $A^u_k(X)$, $A^w_k(X)$, $A^d_k(X)$, and fills in the value one in $A^u_k(X)$ and $A^d_k(X)$ for the entries corresponding to the consecutive vertices in $R_k$, Lines 4 and 6. For the weighted undirected adjacency matrix, the edge weights of the original graph $A^w(X)$ corresponding to the vertices in $R_k$ are filled in, Line 5, and in this way the adjacency matrices corresponding to each obtained route are determined. Note that this approach may be applied for the directed adjacency matrix, because the routes of all transport missions are taken into account separately, and therefore the graph will be traversed only into one direction each time this subalgorithm is executed.

In order to illustrate this approach, a simple graph is introduced, as depicted in Figure 4.1(a). Imagine, that two transport missions exist on this graph, having

\[
\begin{align*}
n^s &= [2 \ 3], & \text{and} \\
n^d &= [9 \ 8].
\end{align*}
\]

(4.1) (4.2)

where $n^s$ contains the starting vertices, and $n^d$ the destination vertices of the different transport missions. Next, Dijkstra’s algorithm, i.e. Algorithm [1] will determine the shortest routes

\[
\begin{align*}
R_1 &= [2 \ 4 \ 6 \ 7 \ 9], & \text{and} \\
R_2 &= [3 \ 4 \ 6 \ 7 \ 8].
\end{align*}
\]

(4.3) (4.4)

Subsequently, Algorithm 2 is executed in order to determine the adjacency matrices corresponding to the obtained routes, of which the unweighted directed adjacency matrices are given by:

\[
\begin{align*}
A^d_1(X) &= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, & A^d_2(X) &= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.
\end{align*}
\]

(4.5)

The correctness of these matrices can be easily verified using the theory presented in 3.2 Algorithm 2 and Figure 4.1(b) wherein route one is coloured in red and route two coloured in blue. Additionally, it can be seen from the figure that the routes form subgraphs of the original supergraph. The undirected adjacency matrices are neglected here for convenience.
Now that the routes are obtained, the platoon routing algorithm has to determine whether vehicles may be able to form a platoon, or not, on the basis of their routes. An approach to determine this, is searching for common vertices in the routes of the different transport missions and by looking in which order the vertices are traversed. Note that there have to be at least two common vertices, because a route can intersect with another route, i.e. routes can share a vertex, without sharing an arc.

However, an easier and more efficient way of implementing is searching for duplicate arcs, from which the direction is immediately found. Additionally, duplicate arcs means that there exist at least two common vertices, and therefore that there exist transport missions which share arcs, i.e. share one or more parts of their route, and may therefore be able to form a platoon. In order to save computational time in the platoon routing algorithm, the transport missions will not be compared pairwise, but a three dimensional matrix, called a multidimensional array, will be created, wherein the obtained directed adjacency matrices of all transport missions are stored in the third direction, called page, see Figure 4.2 for illustration.
From the multidimensional array, duplicate arcs can be easily found by summing up this matrix in the page direction. This approach will be used in Algorithm 3, which takes the unweighted directed adjacency matrices $A^d_k(X)$ of all transport missions $K$ as input and gives a matrix $P$, containing the transport missions which share parts of their route, as output.

**Algorithm 3 Platoon Possibilities**

1: procedure **Platoon Possibilities**($A^d_k(X)$)  
2: $A^d_{tot} \leftarrow \sum_{k=1}^{K} A^d_k$  
3:  
4: $P \leftarrow 0$, $w \leftarrow 0$  
5: for each vertex $i \in V(X)$ do  
6: for each vertex $j \in V(X)$ do  
7: if $A^d_{tot,ij} \geq 2$ then  
8: $w \leftarrow w + 1$  
9: for $k \in \{1, \ldots, K\}$ do  
10: if $A^d_{k,ij} = 1$ then  
11: $P_{wk} \leftarrow k$  
12: end if  
13: end if  
14: end for  
15: end for  
16: end for  
17:  
18: Let $l, m$ be rows in $P$  
19: if for any $w$, $P_{w,l} > 0$ for all $k$ then  
20: $P \leftarrow 1, \ldots, K$  
21: else  
22: if $\exists P_{l,k} = P_{m,k}$ then  
23: $P \leftarrow P_l \cup P_m$  
24: end if  
25: end if  
26: end procedure($P$)

As stated in Line 2, the directed adjacency matrices will be summed in order to create one graph $A^d_{tot}$, containing all arcs of the different transport missions. Then, the algorithm searches for duplicate arcs, i.e. if an entry in the summed matrix $A^d_{tot}$ is bigger than or equal to two, Line 7. The transport missions belonging to the current duplicate arc $ij$ are found in Line 10, and are filled in in the matrix $P$, Line 11, i.e. $P$ will be a matrix containing $w$ rows and $k$ columns, where $w$ equals the number of duplicate arcs. Now that the matrix $P$ contains all duplicate arcs and the corresponding transport missions, the matrix has to be reduced, because only the information regarding which vehicles share parts of their routes is needed.

If there exists any row $w$ in $P$ containing only nonzero values, i.e. if all transport missions share at least one arc, then the algorithm will execute Line 20, resulting in a vector containing all transport missions, because all transport mission have common arcs. However, if there exist separate platoons, i.e. transport missions which share arcs with each other but have no connection to other transport missions, then the algorithm will execute Lines 21-24, resulting in a
reduced matrix $P$ containing the separate platoon possibilities in its rows. This is done by merging rows $l$ and $m$, which contain a common transport missions $k$, Line 23. In order to illustrate this approach, a situation of separate platoons will be provided in Section 5.3.

Considering again the example of Figure 4.1(b), the matrices of (4.5) will be summed up, resulting in

$$A^d_{\text{tot}} = A^d_1(X) + A^d_2(X) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}. \quad (4.6)$$

As can be seen from (4.6), the two transport missions share two arcs, because $A^d_{\text{tot},64} = A^d_{\text{tot},76} = 2$, i.e. the two transport missions may be able to drive in a platoon formation on these two arcs. Additionally, this can be verified from Figure 4.1(b). Hence, Algorithm 3 will find, corresponding to Lines 5-16, the matrix

$$P = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}. \quad (4.7)$$

Next, the algorithm will determine that there exist two rows in $P$ containing only nonzero values, i.e. both transport missions share at least one arc, Line 19. Hence, the algorithm will reduce the matrix $P$ to a vector, containing all transport missions, Line 20, i.e. $P = \begin{bmatrix} 1 & 2 \end{bmatrix}$.

Now that the transport missions which share arcs are determined, the algorithm will merge the matrices of these transport missions in order to obtain one graph, containing only the transport missions which may be able to form a platoon. This will be done using Algorithm 4 by filling in the entries in the adjacency matrices corresponding to the vertices $i$ in $R_k$, where $R_k$ is the route corresponding to the current transport mission $k$ as described before for Algorithm 2.

**Algorithm 4 Merge**

1: procedure MERGE($A^u(X), R_k$)
2:     $A^u_p(X) \leftarrow 0$, $A^w_p(X) \leftarrow 0$, $A^d_p(X) \leftarrow 0$
3:     for each row $q \in P$ do
4:         for each column $k \in P$ do
5:             if $P_{qk} > 0$ then
6:                 for $i = 1, \ldots, r_k - 1$ do
7:                     $A^u_{p,R_k,i,R_k,i+1} \leftarrow 1$, $A^w_{p,R_k,i,R_k,i+1} \leftarrow 1$
8:                     $A^w_{R_k,i,R_k,i+1,R_k,i} \leftarrow A^u_{R_k,i,R_k,i+1}$, $A^w_{R_k,i,R_k,i+1,R_k,i} \leftarrow A^w_{R_k,i,R_k,i+1,R_k,i}$
9:                     $A^d_{p,R_k,i,R_k,i+1} \leftarrow 1$
10:             end for
11:         end if
12:     end for
13: end procedure($A^u_p(X), A^d_p(X), A^w_p(X)$)

21 Platoon coordination and routing
Similar to Algorithm 2, this subalgorithm fills in, in the empty adjacency matrices $A_u^p(X)$ and $A_d^p(X)$, the value one for the entries corresponding to the consecutive vertices in $R_k$, Lines 7 and 9. For the weighted undirected adjacency matrix, the edge weights of the original graph $A_w(X)$ corresponding to the vertices in $R_k$ are filled in, Line 8, and in this way the adjacency matrices corresponding to the platooning transport missions are determined.

In order to illustrate the approach of this subalgorithm the example of Figure 4.1(b) is considered, having the routes of (4.3). The merged unweighted directed matrix as determined by Algorithm 4 will thus be defined by

$$A_d^p(X) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 
\end{bmatrix}, \quad (4.8)$$

which results in the graph as depicted in Figure 4.3.

At this point, the graph has already been simplified to merged subgraphs of the original graph, containing only the vehicles that share parts of their routes. However, the routes of the different transport missions may contain many vertices which are not necessarily needed for the optimization algorithm. As explained in the beginning of this chapter, the routes can be simplified by eliminating vertices and arcs in the routes, such that only the starting and destination vertices, as well as the vertices on which platoons can be formed or split up, remain. The vertices which may be eliminated from the routes are the vertices in between two other vertices, i.e. the vertices having degree two. In order to eliminate these vertices from the routes, Algorithm 5 will be used.
This subalgorithm will eliminate vertices from the routes, such that only the start and destination vertices, having degree one, and the vertices on which platoons can be formed or split up, having at least degree three, remain. The algorithm will use the adjacency matrices $A^u(X)$, $A^w(X)$, and $A^d(X)$, as well as the degree of all vertices $D(X)$, as input.

Algorithm 5 Eliminating Vertices

1: \textbf{procedure} ELIMINATING VERTICES($A^u(X)$, $A^w(X)$, $A^d(X)$, $D(X)$)
2: \hspace{1em} \textbf{for} each vertex $i \in V(X)$ \textbf{do}
3: \hspace{2em} \textbf{if} $D_{ii} = 2$ \textbf{then}
4: \hspace{3em} Let $j, k$ be the neighbours of $i$
5: \hspace{3em} $A^u_{ij} \leftarrow 0$, $A^u_{ji} \leftarrow 0$
6: \hspace{3em} $A^u_{ik} \leftarrow 0$, $A^u_{ki} \leftarrow 0$
7: \hspace{3em} $A^u_{jk} \leftarrow 1$, $A^u_{kj} \leftarrow 1$
8: \hspace{3em} $c \leftarrow A^w_{ji} + A^w_{ik}$
9: \hspace{3em} $A^w_{ij} \leftarrow 0$, $A^w_{ji} \leftarrow 0$
10: \hspace{3em} $A^w_{ik} \leftarrow 0$, $A^w_{ki} \leftarrow 0$
11: \hspace{3em} $A^w_{jk} \leftarrow c$, $A^w_{kj} \leftarrow c$
12: \hspace{3em} \textbf{if} $A^d_{ij} = 1$ and $A^d_{ki} = 1$ \textbf{then}
13: \hspace{4em} $A^d_{kj} \leftarrow 1$
14: \hspace{3em} \textbf{else}
15: \hspace{4em} \textbf{if} $A^d_{ji} = 1$ and $A^d_{kji} = 1$ \textbf{then}
16: \hspace{5em} $A^d_{jk} \leftarrow 1$
17: \hspace{4em} \textbf{end if}
18: \hspace{3em} \textbf{end if}
19: \hspace{2em} \textbf{end if}
20: \hspace{1em} $D_{ii} \leftarrow 0$
21: \textbf{end for}
22: \textbf{end procedure}($A^u(X)$, $A^w(X)$, $A^d(X)$, $D(X)$)

As stated in Line 3, the algorithm will only execute if the current vertex $i$ has degree two. It will then set the entries of the current vertex to zero in all adjacency matrices ($A^u(X)$ Lines 5-6, $A^w(X)$ 10-11, $A^d(X)$ 22-23), and fix the matrices correspondingly. For the unweighted undirected adjacency matrix $A^u(X)$ this is done by setting the entries of the neighbours $j$ and $k$ to one, Line 7, and for the weighted undirected adjacency matrix $A^w(X)$ this is done by setting the entries for the neighbours $j$ and $k$ equal to the sum of weights to the neighbouring vertices $c$, Line 9 and 12, see Figure 4.4 for illustration. In order to fix the unweighted directed adjacency matrix $A^d(X)$, the algorithm will first have to determine in which direction the graph is being traversed. Next, the matrix is fixed by filling in the correct entry corresponding to the determined direction, Lines 14-20. At the end of the subalgorithm, Line 25, the current degree for the eliminated vertex $D_{ii}$ will be set to zero.

23 Platoon coordination and routing
Considering the example of Figure 4.3, Algorithm 5 will determine one vertex having degree two, i.e. vertex six, and will set the entries in (4.8) corresponding to the vertex to be eliminated to zero, i.e. $A^d_{64} = A^d_{67} = 0$ and will fix this matrix correspondingly, i.e. $A^d_{44} = 1$. The new directed adjacency matrix is therefore defined by

$$A^d_p(X) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 
\end{bmatrix}$$

which results in the graph depicted in Figure 4.5. In addition, from the figure can be seen that the edge weight is also fixed correspondingly, where $c = A^w_{36} + A^w_{07} = 3 + 2 = 5$.

Up to now, the routes of the transport missions which share arcs are determined and the vertices having degree two have been eliminated, such that the subgraphs are simplified. However, the optimization algorithm is written in Python, while the platoon routing algorithm is developed...
using Matlab. In order to be able to implement the obtained simplified graph into the optimization algorithm, an interface between the two algorithms is needed. Therefore, at the end of the platoon routing algorithm, the obtained merged matrices have to be converted, such that the optimization algorithm is able to take the matrices as an input, which is done using Algorithm 6. This subalgorithm will convert the weighted undirected adjacency matrix \( A^w(X) \) into a weighted and directed adjacency character string \( \mathcal{G} \), where

\[
\mathcal{G} = \{ R_{k,1} : \{ [x_{R_{k,1}}], [y_{R_{k,1}}], [x_{R_{k,2}}], [y_{R_{k,2}}], \ldots, [x_{R_{k,r_k-1}}], [y_{R_{k,r_k-1}}] \} \},
\]

and \( r_k \) equals the number of vertices in \( R_k \). Furthermore, the subalgorithm will provide the corresponding coordinates matrix \( \text{pos} \), containing the \([x, y]\)-coordinates of all remaining vertices, and routes \( \mathcal{R} \) of the transport missions which may be able to form a platoon, containing the vertices to be traversed:

\[
\text{pos} = \{ R_{k,1} : \{ [x_{R_{k,1}}], [y_{R_{k,1}}], [x_{R_{k,2}}], [y_{R_{k,2}}], \ldots, [x_{R_{k,r_k}}], [y_{R_{k,r_k}}] \} \},
\]

\[
\mathcal{R} = \{ v_1', \ldots, v_{r_k}' \}.
\]

Moreover, at the very end of the platoon routing algorithm, the obtained matrices will be written into a text-file, such that the matrices can be immediately imported into the optimization algorithm.

In order to obtain these character strings, Algorithm 6 takes the merged weighted \( A^w_p(X) \) adjacency matrix, as well as the routes of the platooning transport missions \( R_k \), the matrix containing the platooning possibilities \( P \), and the coordinates matrix \( \text{coords} \), as input. The coordinates matrix contains the \( x\)- and \( y\)-coordinates of all vertices in the road network, stored in the first and second column, respectively.

**Algorithm 6 Interface with Optimization Algorithm**

1: procedure INTERFACE WITH OPTIMIZATION ALGORITHM\( (A^w_p(X), R_k, P, \text{coords}) \)

3: \[ \mathcal{G} \leftarrow \{ \prime \} \]

4: for each row \( q \in P \) do

5: \[ \text{for each column } k \in P \text{ do} \]

6: \[ \text{if } P_{qk} > 0 \text{ then} \]

7: \[ \text{for } i = 1, \ldots, r_k - 1 \text{ do} \]

8: \[ \text{append } \mathcal{G} \text{ with } \{ R_{k,i} : \{ x_{R_{k,i}}, y_{R_{k,i}}, x_{R_{k,i+1}}, y_{R_{k,i+1}} \} \} \]

9: \[ \text{if } R_{k,i} = R_{k,i-1} \text{ then} \]

10: \[ \text{append } \{ R_{k-1,i-1} : \{ x_{R_{k-1,i-1}}, y_{R_{k-1,i-1}}, \ldots, x_{R_{k,i}}, y_{R_{k,i}} \} \} \text{ in } \mathcal{G} \]

11: \[ \text{with } \{ R_{k,i} : \{ x_{R_{k,i}}, y_{R_{k,i}}, x_{R_{k,i+1}}, y_{R_{k,i+1}} \} \} \]

12: \[ \text{end if} \]

13: \[ \text{if } i = r_k \text{ then} \]

14: \[ \text{append } \mathcal{G} \text{ with } \{ R_{k,i} : \{ x_{R_{k,i}}, y_{R_{k,i}}, x_{R_{k,i+1}}, y_{R_{k,i+1}} \} \} \]

15: \[ \text{end if} \]

This subalgorithm basically fills in the characters of the remained vertices of the road network and their corresponding edge weights. This is done, by filling in the vertices \( i \) appearing in the route \( R_k \) for the transport missions in \( P \), i.e. only for the transport missions which may be able to form a platoon.
to form platoons, and their corresponding edge weights $A_{k,i}^w, A_{k,i+1}^w, R_{k,i}$. Note that the optimization algorithm has a different convention compared to the platoon routing algorithm, i.e. the directed adjacency matrix is used the other way around, which means that the row indicates the starting vertex and the column indicates the destination vertex of an arc.

The string $G$ will be repeatedly appended as stated in Line 7, and if the last vertex $r_k$ of $R_k$ is encountered, then the string is closed with an accolade, Lines 12-14. However, if the algorithm encounters a vertex in $R_k$ which also appeared in $R_{k-1}$, which occurs for vertices on which platoons have to split, then $G$ will be appended as stated in Lines 8-11.

In order to illustrate this, the example of Figure 4.5 is considered, where the platooning transport missions are going from vertex two to vertex four having edge weight two, going from vertex three to vertex four having edge weight three, going from vertex four to vertex seven having edge weight five, going from vertex seven to vertex eight having edge weight four, and going from vertex seven to vertex nine having edge weight one, which results in the string
t

"$G = \{(2 : \{4 : 2\}, 3 : \{4 : 3\}, 4 : \{7 : 5\}, 7 : \{8 : 4, 9 : 1\}\}".

As can be seen from the obtained string $G$, two arcs are starting at vertex seven, i.e. vertex seven is the vertex on which the vehicles will split the platoon.

In order to convert the coordinates matrix $coords$ into the character string $pos$, and the route vector $R_k$ into the route string $R$, a similar approach as for $G$ is executed, as given below:

```
16: pos ← ' |
17: for j = 1, . . . , r_k do
18: append pos with 'R_{k,j} : [coords_{R_{k,j},1}, coords_{R_{k,j},2}],'
19: if j = r_k then
20: append pos with 'R_{k,j} : [coords_{R_{k,j},1}, coords_{R_{k,j},2}]'
21: end if
22: end for
23:
24: R ← ' |
25: for j = 1, . . . , r_k do
26: append R with 'R_{k,j},'
27: if j = r_k then
28: append R with 'R_{k,j}'
29: end if
30: end for
31: end if
32: end for
33: end procedure(G, R, pos)
```

The subalgorithm fills in the vertices $j$ of $R_k$, and the corresponding x- and y-coordinates, stored in respectively the first and second column of the coordinates matrix $coords$, as stated in Line 18. If the last vertex $r_k$ in $R_k$ is encountered, the string will be closed with an accolade, Lines 19-21. For the route string, the vertices appearing in $R_k$ are filled in, as stated in Line 26. If the
last vertex \( r_k \) of \( R_k \) is encountered, the string will be closed with a bracket, Lines 27-29.

Now that all subalgorithms are explained, the overall platoon routing algorithm will be summarized, see Algorithm 7. The algorithm takes an undirected adjacency matrix \( A^u(X) \), and starting \( n^s \) and destination \( n^d \) vertices of all transport missions \( K \) as input, and will give the simplified graph \( G \), as well as the corresponding coordinates vector \( pos \) and routes \( R \) of the transport missions which may be to platoon as output, in the format of the optimization algorithm.

Algorithm 7 Platoon Routing
1: procedure PLATOON ROUTING\((A^u(X)/A^d(X), A^w(X), n^s, n^d)\)
2: \hspace{1em} if input is \( A^d(X) \) then
3: \hspace{2em} \( A^u(X) \leftarrow A^d(X) \)
4: \hspace{1em} for each vertex \( i \in V(X) \) do
5: \hspace{2em} \hspace{1em} for each vertex \( j \in V(X) \) do
6: \hspace{3em} \hspace{2em} if \( A^u_{ij}(X) = 1 \) then
7: \hspace{3em} \hspace{3em} \hspace{1em} \( A^u_{ji} \leftarrow 1 \)
8: \hspace{3em} \hspace{2em} end if
9: \hspace{2em} end for
10: end if
11: end if
12: Algorithm 1: Dijkstra
13: Algorithm 2: Adjacency matrices per route
14: Algorithm 3: Platoon Possibilities
15: Let \( N \) be all \( k \) not appearing in \( P \)
16: for each row \( q \in P \) do
17: for \( k \in \{1, \ldots, K\} \) do
18: if any \( P_{qk} > 0 \) then
19: Algorithm 4: Merge
20: end if
21: end for
22: end if
23: end for
24: Algorithm 5: Eliminating Vertices
25: else
26: if \( P = 0 \) or \( N > 0 \) then
27: Algorithm 5: Eliminating Vertices
28: end if
29: end if
30: end for
31: end for
32: Algorithm 6: Interface with Optimization Algorithm
33: end procedure\((G, R, pos)\)

27 Platoon coordination and routing
If however, the input is a directed adjacency matrix, then the algorithm will convert the directed adjacency matrix to an undirected adjacency matrix, as stated in Lines 2-11. This operation has to be executed, because the used implementation of Dijkstra’s algorithm is not able to take directed adjacency matrices as an input [17].

The algorithm will execute the subalgorithms one by one and will merge the matrices, according to the matrix $P$, containing the platooning possibilities. However, if there are transport missions without common arcs, the algorithm will store them into the vector $N$, Line 19, and will keep these transport missions separate from the merged matrices. Additionally, the vertices having degree two will also be eliminated from the routes of the non platooning transport missions, wherein therefore only the starting and destination vertices, having degree one, will remain.

The next chapter will elaborate on different situations which the platoon routing algorithm may encounter, in order to visualize and illustrate the working principles of the platoon routing algorithm.
This chapter will provide a few situations which the platoon routing algorithm may encounter, in order to illustrate the working principles. Therefore, a simplified highway road network of Germany will be used, wherein the vertices denote road junctions/intersections or highway entries/exits, and the edges denote roads in between the vertices. The graph of this simplified highway road network is depicted in Figure 5.1. Note that the presented graph is undirected, i.e. the edges have no direction and transport missions may therefore travel in both directions on the presented roads. In addition, the weights of the edges are unit lengths, where one unit length equals approximately $8.4 \text{ km}$. The input for the platoon routing algorithm will thus be an unweighted undirected adjacency matrix $A^u(X)$, and the weighted undirected adjacency matrix $A^w(X)$ will equal the unweighted undirected adjacency matrix $A^u(X)$.

In addition to the unweighted undirected graph, the algorithm needs starting $n^s$ and destination $n^d$ points of different transport missions. A few combinations of starting and destination points will be presented in the upcoming sections, in order to create and illustrate different situations, where the first combination will result in a platoon possibility.
5.1 Platoon possibility

Imagine that two transport missions exist, one travelling from vertex 1 to vertex 346, and one travelling from vertex 38 to vertex 335, which results in the vectors

\[
\begin{align*}
\mathbf{n}^s &= [1 \quad 38], \\
\mathbf{n}^d &= [346 \quad 335].
\end{align*}
\]

Now that the inputs are given, the platoon routing algorithm will first execute Dijkstra’s algorithm, see Algorithm 1. Dijkstra’s algorithm will find the shortest path between the given starting and destination vertices, and the corresponding matrices are determined using Algorithm 2. This results in the graphs as depicted in Figure 5.2(a), where the original graph is coloured in grey. As can be seen, the routes form subgraphs of the original road network. Furthermore, it can be easily seen, that the two transport missions share multiple arcs.

![Two obtained routes on an undirected road network of Germany](a) Two obtained routes on an undirected road network of Germany

![Two merged and simplified routes](b) Two merged and simplified routes

Figure 5.2: Example of a platoon possibility on a road network of Germany

Algorithm 3 will therefore determine the matrix \( P = [1 \quad 2] \) and hence \( N = [0] \), i.e. both transport missions share parts of their routes. As a result, the matrices corresponding to the routes of the two transport missions will be merged, using Algorithm 4, and the routes depicted in Figure 5.2(a) will form one graph, instead of two different ones.

As can be seen from Figure 5.2(a) the routes contain many vertices having degree two, i.e. vertices in between two other vertices, of which the numbers are also depicted. In order to simplify the routes, Algorithm 5 will be executed. This subalgorithm eliminates all vertices having degree two, such that only the start and destination vertices, having degree one, and the vertices on which platoons can be formed and split up, having degree three, remain. The simplified
directed adjacency matrix results in the graph depicted in Figure 5.2(b).

At the end of the platoon routing algorithm, Algorithm 6 will be executed, resulting in the matrices:

\[ G = \{1 : \{17 : 109.20, 17 \} : \{438 : 369.60, 38 \} : \{17 : 100.80, 346 : 151.20\}\} \]

\[ pos = \{1 : \{498, -78\}, 17 : \{605, -358\}, 38 : \{703, -202\}, 335 : \{432, -1568\}, 346 : \{627, -1555\}, 438 : \{625, -1165\}\} \]

\[ R[1] = [1, 17, 438, 346] \]

\[ R[2] = [38, 17, 438, 335] \]

As can be seen, the unit lengths have been converted to actual lengths of the roads in [km], where e.g. the road from vertex one until vertex 17 equals 109.2 km. Furthermore, it can be seen from the obtained matrices that, indeed only the necessary vertices remain, corresponding to Figure 5.2(b).

The optimization algorithm has been executed for this particular situation, having the starting times \( t^s_1 = 0 \) h and \( t^s_2 = 0.25 \) h, speed constraint \( v_{\text{max}} = 90 \) km/h, air drag reduction coefficient \( \eta = 0.6 \), and having the first transport mission as the lead vehicle of the platoon, as input. Furthermore, the destination time \( t^d \) is calculated using the edge weights in the route in [km] and using a constant speed of 80 km/h. Having these inputs, the optimization algorithm will determine that the vehicles are able to form a platoon, while accounting for the starting times and arrival deadlines, as can be seen from Figure 5.3.

![Figure 5.3: Two platooning transport missions versus time](image-url)
The obtained velocities $V$ (in $[km/h]$) corresponding to the arcs from start to destination vertex are;

- $V[1] = \begin{bmatrix} 72.73 & 82.76 & 79.25 \end{bmatrix}$
- $V[2] = \begin{bmatrix} 80.54 & 82.76 & 74.25 \end{bmatrix}$

From these velocity profiles and from Figure 5.3 it can be concluded that HDVs may travel at higher speeds when driving in platoon formation, which may be beneficial due to a reduced air drag as elucidated in Appendix A. Note, that this is only beneficial due to the time constraints of the transport missions. The optimization algorithm will namely calculate the velocities according to starting times and arrival deadlines, where costs can be reduced by driving slower during individual driving and driving faster during platooning. Additionally, it can be concluded that the first HDV is driving slower than the second one on the first part of its route, in order to form a platoon at the meeting vertex $17$.

The fuel consumption due to air drag as described in Section 2.2 as determined for these two transport missions, for driving in a platoon is reduced by $11.82\%$ compared to individual driving HDVs, where the fuel consumption for individual driving was determined by setting $t_s^2$ much higher than $t_s^1$. Therefore, it can be concluded that forming a platoon of HDVs will reduce the fuel consumption significantly, according to the optimization algorithm.

Additionally, the computational time for the optimization can be measured in order to judge if the simplified graphs enable more efficient computations. The needed computational time using the two mentioned transport missions, and using the routes of the simplified German road network as depicted in Figure 5.2(a) is estimated to be approximately $1.67 \cdot 10^4 s \approx 4.63 h$. This computational time has been estimated by executing parts of the graph and has been extrapolated by fitting an exponential function over the time results. The computational time using the simplified graph as depicted in Figure 5.2(b) equals $0.14 s$. This significant difference can be elucidated by the fact that the simplified graph of Figure 5.2(b) contains less arcs than the graph of Figure 5.2(a), where the computational time enlarges exponentially for increasing number of arcs. From these results, it can be concluded that simplified graphs indeed enable more efficient computations by significantly reducing the computational time.

As explained in Chapter 4, the platoon routing algorithm has been split into subproblems, i.e. the transport missions which may be able to form a platoon are merged to one graph and the non platooning transport missions are kept separate. In order to clarify this, two different situations having no platooning possibility will be highlighted in the next section.

### 5.2 No platoon possibility

Imagine that the same transport missions exist as described in the previous section, but with the second transport mission travelling in opposite direction, resulting in the vectors

$n^s = \begin{bmatrix} 1 & 335 \end{bmatrix}$ \hspace{1cm} (5.3)

$n^d = \begin{bmatrix} 346 & 38 \end{bmatrix}$ \hspace{1cm} (5.4)

The platoon routing algorithm will again determine the graphs depicted in Figure 5.2(a). However, Algorithm 3 will determine that there are common vertices but no duplicate arcs, since the
transport missions are in opposite direction. Subsequently, the matrix \( P = [0] \) and the matrix \( N = [1 \ 2] \), and therefore the algorithm will keep the matrices split. Furthermore, all vertices having degree two are eliminated. Hence, only the starting and destination vertices will remain, see Figure 5.4(a).

![Figure 5.4: Examples of transport missions having no platoon possibility](image)

In addition, if there exist two transport missions travelling in another region on the road network, having

\[
\begin{align*}
\begin{bmatrix} n^s \end{bmatrix} &= \begin{bmatrix} 1 & 161 \end{bmatrix}, \\
\begin{bmatrix} n^d \end{bmatrix} &= \begin{bmatrix} 346 & 265 \end{bmatrix}.
\end{align*}
\]

Algorithm 3 will determine again that the transport missions share no arcs, and will therefore keep the matrices split, see Figure 5.4(b). Obviously, more transport missions on a road network may exist, of which two examples will be given in the next section.

### 5.3 Multiple platoon possibilities

Imagine that there exist three transport missions, having

\[
\begin{align*}
\begin{bmatrix} n^s \end{bmatrix} &= \begin{bmatrix} 1 & 38 & 117 \end{bmatrix}, \\
\begin{bmatrix} n^d \end{bmatrix} &= \begin{bmatrix} 346 & 335 & 296 \end{bmatrix}.
\end{align*}
\]

Algorithm 3 will determine that all three routes share arcs, i.e. \( P = [1 \ 2 \ 3] \) and \( N = [0] \), and will therefore merge all three transport mission to one graph. Subsequently, the routes are
simplified by eliminating vertices having degree two, which results in the graph as depicted in Figure 5.5(a).

![Figure 5.5: Examples of multiple transport missions](image)

Moreover, when there exist five transport missions, resulting in two separate platoons, having starting and destination vertices

\[ n^s = \begin{bmatrix} 1 & 38 & 250 & 117 & 274 \end{bmatrix}, \]
\[ n^d = \begin{bmatrix} 346 & 335 & 186 & 296 & 176 \end{bmatrix}. \]

Algorithm 3 will find duplicate arcs in the matrices corresponding to routes 1, 2 and 3 and routes 4 and 5. Hence, \( N = [0] \) and

\[ P = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 \end{bmatrix}. \]

As described in Chapter 4, the matrix \( P \) will contain the platoon possibilities in its rows, i.e. there exist two separate platoon possibilities. Next, Algorithm 4 will merge the matrices of routes 1, 2 and 3, and the matrices of routes 4 and 5, by filling in the entries in the adjacency matrices corresponding to the routes of the transport missions which may be able to form a platoon in \( P \). Again, the vertices having degree two will be eliminated and the obtained graph is a simplification of the original graph, see Figure 5.5(b).

Besides the examples, many more thinkable situations exist which the algorithm may encounter. The algorithm is also able to handle transport missions having the same starting or same destination vertex, overlapping routes, and multiple transport missions. The next chapter will summarize the conclusions and recommendations of this project.
6 Conclusions and recommendations

As described in Chapter 1, the easy to obtain fossil fuels are becoming more scarce, resulting in increasing fuel prices. Furthermore, the traffic intensity is continuously increasing, resulting in more traffic congestion and a bigger environmental impact, i.e. more fuel consumption and thereby more greenhouse gas emissions. HDV platooning is a developing technology which can help to resolve these issues, as can be concluded from Chapter 1.

In addition, from Appendix A can be concluded that HDV platooning will reduce aerodynamic drag forces, which leads to significant fuel savings as proven for a particular situation in Section 5.1. Additionally, it can be concluded from Appendix A that driving at lower speeds will result in lower air drag forces and thereby a lower fuel consumption. Moreover, from Section 5.1 can be concluded that driving at higher speeds when driving in platoon formation may be beneficial, due to the reduced air drag. Note, that this is only beneficial due to the time constraints of the transport missions. The optimization algorithm will namely calculate the velocities according to starting times and arrival deadlines, where costs can be reduced by driving slower during individual driving and driving faster during platooning.

From Chapter 4 it has become clear that the platoon routing algorithm works as intended, and presents the needed outputs in the correct format, such that the optimization algorithm is able to take them as inputs.

From the test results as stated in Section 5.1 it can be concluded that a simplified graph will result in much lower computational time in the optimization algorithm, which was one of the intentions of determining and simplifying the routes of different transport missions.

The overall conclusion that can be drawn, is that coordination and routing should be applied to form platoons, in order to fully exploit the fuel saving potentials. This can be realised by coordinating scattered vehicles on a road network using intelligent transportation systems, which includes vehicle-to-vehicle and vehicle-to-infrastructure communication. Furthermore, the platoon routing algorithm and the optimization algorithm should work together in order to calculate the velocity profiles on the corresponding routes to obtain fuel-optimal platoons.

Future work on the platoon routing algorithm can be done by connecting the algorithm to a real routing engine, such as Open Street Maps, in order to implement and test more complex or more sophisticated road networks, i.e. road networks containing more vertices and arcs than tested up to now. Moreover, on a bigger road network more separate platoon formations can be created, which might be an interesting thing to test.
Bibliography


37 Platoon coordination and routing


Platoon coordination and routing


A Fuel saving potential

HDV platooning has the potential to provide significant fuel savings, which is mainly caused by a significant reduction in air drag. The aerodynamic force experienced by a moving vehicle can be described as [37]:

\[ F_d = \frac{1}{2} \rho c_d A v^2. \]  (A.1)

In this formula, \( \rho \) denotes the density of air and equals \( 1.293 \ \text{kg/m}^3 \) [53]. \( A \) the vehicle cross section which is commonly \( 9 \ \text{m}^2 \) [39], \( c_d \) the drag coefficient, and \( v \) the vehicle’s velocity in \([\text{m/s}]\).

The aerodynamic drag can amount up to 50% of the total resistive forces experienced by a HDV at full speed and is therefore of much importance. Different studies, i.a. on Computational Fluid Dynamics (CFD), have demonstrated that the aerodynamic drag can be significantly reduced by arranging trucks in a platoon formation. As can be seen from Figure A.1, especially the follower vehicles in a platoon will experience a reduced relative wind speed and hence an overall reduced air drag force. The wind speed continues to diminish as the inter-vehicular distance decreases. In addition, the lead vehicle of the platoon may also experience a reduction in air drag if the relative distance to the follower vehicle is small enough to diminish the turbulence which occurs behind the lead vehicle [3] [34] [40].

![Figure A.1: CFD results for two HDVs driving at varying inter-vehicular distances](image-url)
Additionally, empirical results with three HDVs, which are derived through measurements in a wind tunnel, show a significant reduction in air drag coefficient \( c_d \), as depicted in Figure A.2. As can be seen, all vehicles will experience a lowered air drag coefficient with reducing inter-vehicular distance. Moreover, it can be concluded that especially the following vehicles will experience a reduction in air drag coefficient [3] [34].

![Figure A.2: Empirical results for the change in air drag coefficient with varying inter-vehicular distance [3]. Adapted by Alam from Hucho [30]](image)

Furthermore, experimental results, see also Figure A.2, show that the drag coefficient decreases with increasing order in a platoon. Hence, the aerodynamic effects may be transferred from vehicle to vehicle and the air drag is reduced further with increasing vehicle position in a long and structural platoon. Additionally, a small vehicle following a large vehicle will gain even more benefit in the reduction of air drag [10] [55].

Another important force is the force related to rolling resistance, which is given by [37]:

\[
F_r = m g c_r \cos(\alpha),
\]

(A.2)

where \( m \) denotes the Gross Vehicle Weight (GVW) in [kg], \( g \) the gravitational acceleration which equals 9.81 \( m/s^2 \), \( c_r \) the rolling resistance coefficient, which is 0.006 - 0.01 for truck tyres on asphalt [20], and \( \alpha \) the road gradient in [°].

Considering a tractor-semitrailer driving on a flat road (\( \alpha = 0° \)), having a GVW of 40 tonnes, assuming a constant \( c_d \) of 0.7 [30], a constant \( c_r \) of 0.008 [20], and using the formulae of the forces in (A.1) and (A.2), the graph depicted in Figure A.3 can be obtained. From this figure
and from (A.1), it can be concluded that the aerodynamic force increases quadratically with the speed, and will therefore become bigger than the rolling resistance. This implies that driving at low speeds will lead to a low aerodynamic force. However, it may be beneficial to drive faster while driving in platoon formation, compared to individual driving, due to a reduced aerodynamic drag [51], see Section 5.1 for a more detailed elucidation.

Figure A.3: Aerodynamic and rolling resistance force as function of velocity