University of Bremen

Doctoral Thesis

Decision support in international long-haul freight transportation for the planning of rest periods, breaks and vehicle refueling

Author: Alexandra Bernhardt

Reviewers: Prof. Dr. Herbert Kopfer  
Prof. Dr. Teresa Melo

A dissertation submitted to the
Doctoral Commission of
Faculty 7: Business Studies & Economics
in fulfillment of the requirements for the degree of Dr. rer. pol.

April 24, 2019
Abstract

In international long-haul freight transportation, truck drivers are often on the road for several consecutive days or even weeks. During their trips, they must comply with the rules on driving time, breaks and rest periods which in the European Union are governed by Regulation (EC) No 561/2006 which entered into force in April 2007. As the regulation has a high influence on the transport durations, it has to be taken into account when planning arrival times at customer locations and choosing among multiple customer time windows. Considering transport costs, a high attention should be paid to fuel costs as fuel is one main cost driver in the road haulage sector. An analysis of diesel price variations across different European countries showed that a significant potential for cutting fuel expenditure can be found in international long-haul freight transportation. Thus, fuel costs and driver rest periods and breaks are two important issues that transport companies have to take into account to be profitable. Dependencies among the corresponding planning tasks suggest a joint consideration. In this thesis, the resulting problem is approached gradually by starting with the isolated consideration of the two subproblems.

For the planning of rest periods, breaks and customer time windows, two mixed integer linear programming (MILP) models and solution strategies are proposed. Together with a transformation algorithm they allow to plan driver activities in compliance with Regulation (EC) No 561/2006 for a given sequence of customer locations and other stops to be visited by a vehicle. One of the models considers all rules, including extended rules, while the other takes into account the regular requirements. A special feature is the consideration of "soft" time windows which has not been studied in this context so far. In addition to the mathematical models, a myopic algorithm was developed that can only "see" the route until the next customer stop and the corresponding customer time windows in advance and plans driver activities accordingly. The advantages of the different approaches are evaluated.

The refueling subproblem is addressed by extending the standard fuel optimizer model presented by Suzuki (2008, 2009) which takes into account detours to reach gas stations with attractive fuel prices. Additionally to the original version, the consideration of time windows is included. In a short digression, the new MILP model is embedded in the insertion heuristic developed by Solomon (1987) for solving the vehicle routing problem with time windows (VRPTW).

A joint consideration of rest periods, breaks and refueling is achieved by merging the MILP models developed for the isolated problems to one model. The solution process for the resulting multicriteria optimization problem with the goals to minimize lateness, completion time and fuel expenditures is described. Additionally, a preprocessing heuristic is proposed which reduces the number of gas stations to be considered along the route of a vehicle and thus the solution space and the computational effort.

For each of the models and algorithms presented numerical experiments were conducted. For the extended VRPTW, the well-known Solomon benchmark instances were modified. The other experiments were performed with instances derived from real-world data that include vehicle routes for one week and information on gas stations along the vehicle routes.

For future research, the main elements are proposed that together with metaheuristic strategies can be used to develop a heuristic for the combined problem.
Acknowledgements

I am grateful to all of those who supported me during writing this thesis. I thank all of them, offering my special thanks to the ones below.

First, I would like to thank my advisor Prof. Dr. Herbert Kopfer for his support and his patient guidance throughout this thesis which encouraged me finishing this work.

I thank my supervisors Prof. Dr. Teresa Melo and Prof. Dr. Thomas Bousonville for their support that extended well beyond the DynaServ project from which the idea of this thesis arose. Special gratitude I would like to express to Prof. Dr. Teresa Melo for reading all of my manuscripts and her numerous comments and useful critiques on this research work. Prof. Dr. Bousonville contributed with many ideas to possible contents giving this thesis and particularly the test instances a practical orientation.

Next, I wish to thank my former colleagues at ISCOM, Oliver Bindel and Martin Dirichs, for the cooperation developing the DynaServ prototype enabling me to work with real-world test data.

Finally, I thank my family, friends and colleagues for their mental support. I would like to express my very special thanks to my beloved husband Christian who encouraged me throughout writing this thesis and always supported me.

Alexandra Bernhardt
Kindsbach, August 2018
## Contents

Abstract

Acknowledgements

1. Introduction
   1.1. Research motivation
   1.2. Research objectives
      1.2.1. Distributed decision making - A short process analysis
      1.2.2. Problem description
      1.2.3. Possible integration of the developed models and algorithms into a decision support system
   1.3. Outline of the thesis

2. Scheduling of driving times, breaks and rest periods
   2.1. Problem description
   2.2. Outline
   2.3. Regulation (EC) No 561/2006
   2.4. Literature review
   2.5. Mathematical formulation
      2.5.1. Modeling techniques - A digression on modeling logical conditions with binary variables
      2.5.2. Parameters of the model
      2.5.3. Variables
      2.5.4. Optional rules
      2.5.5. Begin of service constraints
      2.5.6. Time window constraints
      2.5.7. Lateness constraints
      2.5.8. Maximum time between two consecutive weekly rest periods
      2.5.9. Durations of daily rest periods
      2.5.10. Indicator variables for daily rest periods on arcs
      2.5.11. Indicator variables for breaks on arcs
      2.5.12. Decision variables that indicate a necessary break
      2.5.13. Decision variables that indicate that a break has already been taken
      2.5.14. Indicator variables for early daily rest periods
      2.5.15. Vertex activity constraints
      2.5.16. Get status constraints
4.8.3. Managerial insights .......................................................... 205
4.9. Possible Heuristic Approaches .................................................. 210

5. Summary and future research ......................................................... 217
   5.1. Summary ............................................................................ 217
   5.2. Future research ................................................................. 221

Appendices ................................................................................. 223

A. Parameters and variables of the combined MILP model .................. 225
   A.1. Parameters of the MILP model ............................................. 225
   A.2. Variables of the MILP model ............................................... 228

B. Detailed results of numerical experiments for the combined problem ........ 235

C. Heuristic Approaches - Pseudo code ........................................... 243

Bibliography .............................................................................. 257
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Cost breakdowns of hauliers from selected EU member states</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Price difference between German cities on February 12, 2018</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Diesel prices excluding VAT in Euro per liter on December 9, 2017</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Diesel prices excluding VAT in Euro per liter on February 13, 2018</td>
<td>4</td>
</tr>
<tr>
<td>1.5</td>
<td>Planning tasks</td>
<td>6</td>
</tr>
<tr>
<td>1.6</td>
<td>Planning of rest periods and breaks without considering refueling</td>
<td>8</td>
</tr>
<tr>
<td>1.7</td>
<td>Symbols describing driver activities</td>
<td>8</td>
</tr>
<tr>
<td>1.8</td>
<td>Joint planning of rest periods, breaks and refueling</td>
<td>10</td>
</tr>
<tr>
<td>1.9</td>
<td>Service oriented architecture</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>Relation of the different time horizons (Meyer and Kopfer (2008))</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of a sequence of vehicle stops over a time horizon of one week</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>Vertices</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>Driver time management activities in a vertex</td>
<td>26</td>
</tr>
<tr>
<td>2.5</td>
<td>Driver time management activities on an arc</td>
<td>26</td>
</tr>
<tr>
<td>2.6</td>
<td>Partial schedule with vs. without early daily rest period</td>
<td>28</td>
</tr>
<tr>
<td>2.7</td>
<td>Partial schedule with vs. without a break in vertex i</td>
<td>28</td>
</tr>
<tr>
<td>2.8</td>
<td>Splitting daily rest periods</td>
<td>43</td>
</tr>
<tr>
<td>2.9</td>
<td>The impact of an early daily rest period ($\mu_{(1,2)}^{\text{early}} = 1$)</td>
<td>62</td>
</tr>
<tr>
<td>2.10</td>
<td>Transformation is necessary</td>
<td>87</td>
</tr>
<tr>
<td>2.11</td>
<td>Fitting the computed route to the driver's route</td>
<td>101</td>
</tr>
<tr>
<td>2.12</td>
<td>Solution processes with and without an upper cutoff</td>
<td>103</td>
</tr>
<tr>
<td>2.13</td>
<td>Run times for the MILP model with optional rules</td>
<td>106</td>
</tr>
<tr>
<td>2.14</td>
<td>Average run time depending on the number of vertices</td>
<td>107</td>
</tr>
<tr>
<td>2.15</td>
<td>The influence of the number of time windows (TW) on the run time</td>
<td>108</td>
</tr>
<tr>
<td>2.16</td>
<td>The influence of the time window length on the run time</td>
<td>109</td>
</tr>
<tr>
<td>2.17</td>
<td>The influence of the time window length on the run time</td>
<td>110</td>
</tr>
<tr>
<td>2.18</td>
<td>Average run times <strong>with</strong> and <strong>without</strong> considering optional rules</td>
<td>112</td>
</tr>
<tr>
<td>2.19</td>
<td>Average total lateness with and without considering optional rules</td>
<td>113</td>
</tr>
<tr>
<td>2.20</td>
<td>Average schedule duration with and without considering optional rules</td>
<td>114</td>
</tr>
<tr>
<td>2.21</td>
<td>Myopic heuristic - flowchart</td>
<td>116</td>
</tr>
<tr>
<td>2.22</td>
<td>Scheduling activities on arc $(i, i + 1)$</td>
<td>121</td>
</tr>
<tr>
<td>2.23</td>
<td>Choose time window at stop $i + 1$</td>
<td>125</td>
</tr>
<tr>
<td>2.24</td>
<td>Modify rest durations and schedule activities at stop $i + 1$</td>
<td>130</td>
</tr>
<tr>
<td>2.25</td>
<td>Route of base instance 3</td>
<td>133</td>
</tr>
<tr>
<td>2.26</td>
<td>Average lateness of schedules depending on the solution technique</td>
<td>138</td>
</tr>
<tr>
<td>2.27</td>
<td>Average schedule duration depending on the solution technique</td>
<td>139</td>
</tr>
<tr>
<td>2.28</td>
<td>Impact of time windows on lateness</td>
<td>140</td>
</tr>
</tbody>
</table>
4.17. Lateness and refueling cost depending on the filter distance .............. 206
4.18. Average completion time and filter distance .................................. 207
4.19. Graph structure for the extended myopic algorithm ......................... 211
4.20. Graph structure for the refueling subproblem (gas stations are represented by 1,2,3,...) ................................................................. 212
4.21. Algorithm for the fixed route refueling problem respecting the minimum purchase quantity ................................................................. 214
List of Tables

2.1. Characteristics of the test instances ............................................. 102
2.2. Total number of variables and constraints in the MILP model with optional rules (3 alternative time windows) ............................................. 102
2.3. Run times in seconds for the MILP model without additional step for an upper cutoff ................................................................. 104
2.4. Run times in seconds for the MILP model with additional step for an upper cutoff ................................................................. 105
2.5. Run times in seconds for the MILP model without optional rules ................................................................. 111
2.6. Time windows ........................................................................... 134
2.7. Optimal schedule identified by the MILP model without optional rules ................................................................. 135
2.8. Schedule created with the myopic heuristic ................................................................. 136
2.9. Optimal schedule identified by the MILP model with optional rules ................................................................. 137
3.1. Input parameters ....................................................................... 154
3.2. Decision variables ..................................................................... 154
3.3. Computational results for the 6 parameter sets; each line sums up the results of the 56 runs ............................................................................. 166
3.4. Relative increase of the tour length when comparing a spread of 10%, respectively 20% to a constant price structure (no variations) ................................................................. 166
3.5. Discarded inserts of customer locations due to having reached the time limit to obtain a feasible refueling solution ................................................................. 168
4.1. Base instances ........................................................................... 180
4.2. Remaining locations after filtering ..................................................... 189
4.3. Average number of variables and constraints (one time window) depending on the filter distance ............................................................................. 190
4.4. Time windows ........................................................................... 191
4.5. Schedule from optimization step 1 ..................................................... 194
4.6. Schedule from optimization step 2 ..................................................... 195
4.7. Schedule from optimization step 3 ..................................................... 196
4.8. Schedule from optimization step 4 ..................................................... 197
4.9. Solution quality and run times depending on the filter distance ................................................................. 207
4.10. Filter distance and average detour distance ................................................................. 208
4.11. Filter distance and the number of refueling stops ................................................................. 209
4.12. The number of time windows and the number of refueling stops ................................................................. 209
4.13. Time window length and the number of refueling stops ................................................................. 209
B.1. Filter distance 100 km: Run times in seconds for the MILP model solution process ................................................................. 236
B.2. Filter distance 200 km: Run times in seconds for the MILP model solution process ................................................. 237
B.3. Filter distance 300 km: Run times in seconds for the MILP model solution process ................................................. 238
B.4. Filter distance 400 km: Run times in seconds for the MILP model solution process ................................................. 239
B.5. Filter distance 500 km: Run times in seconds for the MILP model solution process ................................................. 240
B.6. Filter distance 1000 km: Run times in seconds for the MILP model solution process ................................................. 241
1. Introduction

Increasing just-in-time management practices, growing pressure on satisfying customer demands on time, and the need to keep transport costs low put high pressure on truck drivers, dispatchers, and their transport companies. When planning arrival times of long-haul trips that require several days, Regulation (EC) No 561/2006 concerning driving and working hours of drivers in road transport is obligatory in all member countries of the European Union (EU). Considering transport costs, special attention should be paid to fuel costs as fuel is one of the two main cost drivers in the road haulage sector. Thus, fuel costs and driver rest periods and breaks are two important issues that transport companies have to take into account to be profitable. Dependencies among the corresponding planning tasks suggest a joint consideration.

Technology such as on-board computers, digital tachographs, and telematics equipment opens many new opportunities for transport companies. Online available telematics data such as latest position data of vehicles and time management data which reflect the exact status considering rest periods and breaks of drivers, gas station prices and locations, are some of the data which could help dispatchers and drivers in their daily work. As there is plenty of distributed data to be evaluated decision support systems with advanced planning tools can be an important contribution to support decision-makers.

This chapter is organized as follows. Section 1.1 describes the research motivation. In Section 1.2 the research objectives are presented. Section 1.3 gives an outline of this thesis.

1.1. Research motivation

Considering the competitiveness in the road haulage sector, cost levels are a key factor. According to the European Commission (2014), converging cost structures will more and more urge transport undertakings to improve their efficiency and quality of service. As depicted in Figure 1.1, fuel is a main cost driver, representing between 24% and 38% of the total costs in the EU member states.¹

¹ Source: Collection and Analysis of Data on the Structure of the Road Haulage Sector in the European Union, AECOM 2013 (retrieved from European Commission (2014)).
An analysis of the mean prices of the 100 largest cities in Germany reveals that price variation may amount up to 13.6 ct per liter diesel considering the lowest and the highest price (see Figure 1.2),\(^2\) yielding a variation of about 14%. Since average values are considered, prices may well vary more than 14%.

\(^2\) Source: clever-tanken.de (2018). The referenced prices are given excluding VAT.
1.1. Research motivation

There seems to be a high cost saving potential but many haulage companies have contracts with fuel card operators that make diesel prices at gas stations dependent on list prices per country. In that case, price variations within one country become less important. A significant potential for cutting fuel expenditure can especially be found in international long-haul freight transportation. In the European Union, international transport operations account for almost one third of all road freight transport activities (European Commission (2014)). Diesel prices vary strongly across different European countries. Variations may amount to 30 ct per liter and more (see Figure 1.3\(^3\)). Price relations between countries are not constant over time as the comparison of Figures 1.3 and 1.4\(^4\) shows. This means that even if countries with cheap diesel prices were chosen for refueling in the past and fixed contracts exist causing traveled routes to remain the same, new refueling plans are necessary to be developed on a regular basis to exploit the cost saving potential.

Not only the choice of gas stations but also the refueling quantities have an impact on the refueling costs of a trip and determining good refueling strategies is a non-trivial task. Additional cost or non-cost factors can be taken into account and are considered in the literature (see Section 3.3). One important factor is the time needed for detours to gas stations and for refueling, and its impact on driver schedules.

Besides cost levels, quality of service is another important key factor (European Commission (2014)) for which in the haulage sector punctuality is a quantifiable distinctive feature for the customer. Lateness may cause contractual penalties and may lower customer satisfaction which has a strong impact on future requests and thus on the economic viability of a haulage company. The time needed for detours and refueling influences the transport duration and should be taken into account. In particular, cheap refueling may bear the risk of a late arrival and in this case the refueling plan should be reconsidered.

The cost breakdowns in Figure 1.1 depict that labor is, with fuel, one of the two main cost drivers in the road haulage sector. Opportunity costs arise when a driver has to wait several hours for a new customer time window or until the next day to load and/or unload the vehicle because he missed a time window or arrived after the opening hours. If the driver arrives much too early, this is disadvantageous as well for the same reason. Finally, one should not forget the annoyance for the driver if, for example, deviations from the original schedule disturb his or her plans for the weekend or a resting location with basic amenities.

When planning arrival times at customers for long-haul trips that require several days, Regulation (EC) No 561/2006 on driving times, breaks, and rest periods of drivers in road transport is obligatory in all member countries of the EU. The corresponding rules were devised to improve safety and working conditions of drivers in road transport, and have a high influence on the execution time of a transport request. Disregarding them may be fined severely.

Despite the rules being rather complex in their application, as often many different possibilities to plan driver activities have to be evaluated, a dispatcher has to set up his plans ensuring that drivers are able to stick strictly to the regulation. Rest periods and breaks cannot be split arbitrarily or interrupted to serve customers or to refuel.

\(^3\) Source: Europe’s Energy Portal (2017)
Figure 1.3.: Diesel prices excluding VAT in Euro per liter on December 9, 2017

Figure 1.4.: Diesel prices excluding VAT in Euro per liter on February 13, 2018
Thus, it is not recommendable to consider resting activities in the form of a fixed proportion of the overall travel time in the schedule when planning arrival times as deviations would occur frequently. This would be disadvantageous especially if narrow time windows are involved.

1.2. Research objectives

In this section, the research objectives of this thesis are described. In the first part (Section 1.2.1) we present the findings of a process analysis conducted on site at a haulage company to give the practical context for the problem description that follows in Section 1.2.2. In Section 1.2.3, the possible integration of the developed models and algorithms into a decision support system is discussed.

1.2.1. Distributed decision making - A short process analysis

During our research, we cooperated with a medium-sized company operating transport services in Europe. An on-site process analysis gave detailed insights into the planning process and the distribution of responsibilities. Following Bousonville et al. (2012), Figure 1.5 shows the planning tasks that are relevant for our consideration. The tasks are distributed among dispatchers (tasks 1 to 7), a decision maker for refueling matters (tasks 8 and 9) and drivers (tasks 10 and 11). The solid arrows show the temporal and logical sequence of decisions.

We concentrated on the international transport requests which represent a large proportion of the business activities of the freight company. Each transport request consists of a pickup and a corresponding delivery location. These locations are often far apart and only a few customers can be serviced during the same week. Often, the fulfillment of a transport request extends to the following week. Then, the driver does not return home but takes his weekly rest period somewhere near to the route. Many drivers return home only after several weeks. For the arrival at customer locations, opening hours have to be taken into consideration. Several large customers propose each working day multiple alternative time windows, i.e. time intervals in which loading and/or unloading should take place. A choice among the time windows has to be made by the dispatcher. Especially for this group of requests, the joint consideration of driver rest periods and breaks and refueling is promising. As national borders are passed and fuel prices vary considerably across different countries in Europe, there is a high potential to cut fuel expenditures. Since travel times of several days are considered, the integration of rest periods and breaks into the planning of arrival times and the choice of time windows must not be neglected.

Time windows are very common in long-haul freight transportation. Favorable time windows may be occupied if not chosen early enough but a reliable estimation of arrival time is necessary. Therefore, time windows are not necessarily scheduled when accepting transport
requests, they are chosen when the arrival time at the customer location is predictable, i.e. one or two days in advance, depending on the demand.

While a significant part of transport requests result from fixed contracts, others are not known in advance. Several partner companies pass on requests and additionally, the dispatchers use online freight exchanges to supplement partial loads or to acquire additional requests for the return trip. Conversely, they pass on shipping orders to subcontractors.

In a manual planning phase, a dispatcher clusters transport requests, assigns them to vehicles, and determines the sequence in which customer locations shall be visited. Since only a few of the transport requests are known at the start of the week, additional requests are accepted and assigned to vehicles as the time unfolds. Furthermore, dispatchers are responsible to plan vehicle routes, plan arrival times, and choose among customer time windows. Avoiding lateness is essential in this context. The estimated duration of travel between locations includes a time buffer for rest periods and breaks that is proportional to the distance traveled. Arrival times at customer locations are planned accordingly and drivers are informed.
Drivers are responsible for planning their rest periods and breaks, and regularly inform the dispatcher about their status. The dispatcher uses this information for the planning of time windows and to react to unforeseen events. The decision maker for refueling matters analyzes refueling data from the past and negotiates contracts with fuel card operators. If a gas station is favorable because of its geographical position near to often traveled routes, the decision maker may try to negotiate a special price and informs the dispatchers accordingly. Moreover, the tasks include the monitoring of diesel list prices of various countries and advice is given to the dispatchers about countries where refueling should preferably take place. The dispatchers inform the drivers about changes in the refueling strategy that are relevant for the current route. When refueling is necessary, drivers preferably stop at gas stations with negotiated special price or choose a gas station according to the refueling strategy. Drivers fill up completely if no cheaper gas stations are expected to be passed until the next refueling stop. Otherwise, a smaller amount is refueled.

As information is distributed among the actors, feedback loops (dashed arrows between activities in Figure 1.5) are necessary during the planning process. Additionally, all actors have to handle the stochastic environment inherent in the planning process. This is illustrated by the dashed arrows from the stochastic environment and unforeseen events to the corresponding activities in Figure 1.5. At the start of the week orders are not all known in advance. Many unforeseen events such as order cancellations or changes, traffic jams, fuel price changes, problems with overcrowded rest areas, delays at customers, wrong freight loaded, etc. may occur. Feedback loops and the stochastic environment trigger necessary re-planning actions.

### 1.2.2. Problem description

We assume that the initial clustering and assignment of requests to drivers and vehicles have already been done and concentrate on the planning tasks for a single driver and vehicle. The sequence of customer locations that are assigned in the current week and the route to be traveled are given. Geographical positions of gas stations along the route and the corresponding diesel prices as well as the current fuel level in the tank are known. Driving durations and fuel consumptions between consecutive customer locations are additional input parameters as well as the current time and the driver status concerning rest periods and breaks. As described in the previous section, re-planning will be necessary during the week. The driver status that can be determined at any point in time considering the driver activities since the start of the week and its integration into the input parameters allows an online re-planning.

As we consider a planning horizon that comprises several days, we also take Regulation (EC) No 561/2006 on driving times, breaks and rest periods of drivers into consideration. For each customer location there may be one or more time windows among which a choice has to be made. The time that is needed for loading, unloading and handling activities at each customer location is given as well.

The objective is to optimally choose customer time windows and gas stations, plan refueling amounts and schedule driver activities including refueling so as to maximize punctuality,
i.e. to minimize lateness (soft time windows), and to minimize fuel costs. Inefficiencies that arise from the distributed decision making of drivers and dispatchers will be minimized by considering these tasks simultaneously. Minimizing lateness and minimizing fuel costs are two conflicting goals. For example, choosing a very cheap gas station may cause a greater detour distance and travel duration. This may lead to lateness at following customer locations that could have been avoided by choosing a different gas station for refueling that is more expensive but causes less detour duration.

Figure 1.6 shows an example of the interdependencies of planning refueling and choosing among customer time windows, thereby considering daily rest periods and breaks. The symbols that are used to describe driver activities are explained in Figure 1.7.
It is assumed that the driver crosses two country borders and thus has the possibility to refuel in three different countries with three different list prices. Due to the remaining fuel quantity in the tank, a "corridor" has been determined in which the next refueling has to take place such that the vehicle does not run out of fuel. Without considering time windows, a refueling plan would recommend to stop in the area where the fuel price is 1.10 € and completely refill or at least refuel as much as needed to cross the area with a fuel price of 1.14 €. The latter recommendation depends on the development of future fuel prices in the following regions (countries) to be passed, namely if they are expected to be cheaper or not.

At the bottom of Figure 1.6 driver activities were planned without considering the duration for refueling. The start and the end times of the chosen customer time windows are marked with bold, black vertical lines. Loading and/or unloading at a customer location has to start within a time window but can be finished after the upper bound of that time window. The red vertical lines show the estimated arrival times of the driver at the different customer locations.

It can be seen that there is no lateness and at the first and last customer locations, the driver even has to wait 30 minutes before service can begin. At the second customer location, loading and/or unloading is planned to start at the end of the time window and thus any delays between the first and the second customer would cause lateness.

In this simple example, the estimated duration for refueling is assumed to be 20 minutes, the travel times for detours to gas stations are neglected. If we try to bring the two plans together, the first one only made for refueling, the other one for driver activities (without refueling) and time windows (see the dashed lines from top to bottom), we see that refueling for the optimal price of 1.10 € would cause lateness at customer 2. If we consider punctuality to be more important than fuel costs, refueling (at least) has to take place in the corridor with a diesel price of 1.17 € if the re-planning of time windows, rest periods and breaks is not to be considered.

Figure 1.8 shows the advantages of an alternative planning technique which simultaneously considers the choice of time windows and the determination of driver activities together with refueling. For the second customer a different time window has been chosen thus allowing the driver to be on time and also making it possible to refuel for the cheapest price of 1.10 €. Refueling is depicted in yellow.

The integration of both issues - the scheduling of drivers' rest periods and breaks, and the refueling planning - into one planning process has not been addressed in the literature so far. In this thesis, we will deal with the mathematical modeling and the development of algorithms to integrate the two issues described above into one solution process.

---

5 For reasons of simplicity we assume that the gas station prices in a country are equal to the list price of this country.

6 Note that this is just a modeling decision. If in reality loading and/or unloading should be finished within the time window, subtract its duration from the upper bound of the time window to obtain the corresponding upper bound for the model input.

7 Note that this is a simple example. In reality, if the driver schedule is executed, the complete plan of driver activities may have to be reconsidered after the first refueling. Because of detours, breaks may have to be taken earlier and because of the additional time required, daily rest periods may have to be rescheduled. This is also the case if gas station prices are neglected.
1.2.3. Possible integration of the developed models and algorithms into a decision support system

The idea of the theme of this thesis arose from the project DynaServ\textsuperscript{8} which aimed at integrating data from heterogeneous and distributed systems to improve decision making in international truck load dispatching. Relevant data can, for example, be acquired from on-board computers, digital tachographs and telematics equipment. Most of the data are available online and comprise latest vehicle positions, time management data, gas station prices, and locations. Bousonville et al. (2012) describe the service-oriented architecture (SOA) which was pursued during the project when implementing the corresponding interfaces of the DynaServ prototype in which the models and algorithms that are developed in this thesis can be embedded.\textsuperscript{9} Figure 1.9 gives an overview of the data provided by different sources and a possible arrangement of services for the decision support system.

The fleet management system (FMS) is a standardized CAN bus interface for commercial vehicles with the aim to monitor vehicle parameters such as vehicle speed, mileage, axle load, current fuel level, fuel consumption, and tachograph data. With the help of the GPS, the latest vehicle positions can be identified. Fleet telematics systems allow the transmission of the vehicle related data which in turn can be provided by a corresponding fleet management software via web services. In this way, the fleet management software

\textsuperscript{8} The project was supported by the German Federal Ministry of Education and Research (BMBF) for three years (2009-2012).

\textsuperscript{9} For more details on the implementation of the architecture see Bousonville et al. (2012).
of the haulage company that was a cooperation partner in the DynaServ project was connected.

Information about locations of gas stations and daily fuel prices were provided to the haulage company as downloadable spreadsheets via a web portal with private access by one of the oil companies. For another oil company, PDF-documents with updated price information were received on a regular basis.

Geographical information about the street network was retrieved from OpenStreetMap, a crowdsourcing initiative. Map data of commercial providers such as Google Maps, Bing Maps, HERE or TomTom may alternatively be incorporated.

The software provider of the order management system was a cooperation partner in the DynaServ project. Transport order information such as customer locations, the sequencing of customer locations and initially planned arrival times at customer locations were transferred to the fleet management software via an interface such that in our case such data were available via the web services mentioned above. A direct connection to the decision support system is desirable in practice to be able to transfer updated planning information such as planned arrival times.

Several customers provide online platforms which allow to book time windows for loading and/or unloading. The usage of the corresponding interfaces would allow direct data interchanges and reduce the effort for data administration in different software systems.

Even though it is possible for some basic services to define service interfaces retrieving all data to complete a service call from remote systems, this may lead to a significant performance reduction. Thus, it is better to store most of the data in an own database and to provide necessary update mechanisms.

Figure 1.9 shows a possible service structure with the classification scheme provided by Bousonville et al. (2012). Services are categorized as information, knowledge or business services based on their purpose and their dependency on other services. Information services serve to cluster the heterogeneous raw data originating from different sources and to provide them domain specific. Knowledge services process the data accessed via the information services in an algorithmic way to obtain aggregated information. Business services make use of information and knowledge services to provide decision support to the dispatcher.

In this thesis, we present models and algorithms to choose appropriate customer time windows, determine arrival times, plan rest periods and breaks according to Regulation (EC) No 561/2006 and to derive an efficient refueling plan. Embedded within a business service as shown in Figure 1.9 they can be an important value added within a decision support system as they can help dispatchers to plan vehicle movements, to avoid or prematurely detect lateness and to reduce fuel expenditures.

Possible algorithms for the depicted Gas Station Preselection Service (knowledge service) are presented as well. Why preselections are necessary and how this can be done appropriately will be described in Chapter 4.
Figure 1.9.: Service oriented architecture
1.3. Outline of the thesis

We gradually approach the problem described in Section 1.2.2 by starting with the consideration of two subproblems. In Chapter 2, we concentrate on the scheduling of driver activities in accordance to Regulation (EC) No 561/2006 considering a given sequence of customer locations with multiple customer time windows pursuing the goals of minimizing lateness and overall schedule duration. The chapter starts with a description of the subproblem (Section 2.1) and, after a short outline (Section 2.2), the rules of Regulation (EC) No 561/2006 (Section 2.3). Then a review of related literature is given (Section 2.4). After a short digression on modeling techniques, two MILP models and the corresponding optimization strategies are developed that, together with a transformation algorithm, allow to plan driver activities in compliance with the regulation (Sections 2.5 and 2.6). One of the models considers all rules, including extended rules, while the other takes into account the regular requirements. A special feature is the consideration of "soft" time windows which has not been studied in this context so far. If time windows cannot be met, the resulting schedule gives important information to the dispatcher that is necessary to set up a better schedule. In online re-planning, lateness can be revealed at an early stage such that it is possible to reorganize the schedule or to negotiate new arrival times with customers before communication effort and costs increase, and further delays or cancellations are unavoidable. The numerical results obtained with an off-the-shelf commercial solver are presented in Section 2.7. Test instances were derived from real data and include vehicle routes for one week. In addition to the mathematical models, a myopic heuristic is presented that can only "see" the route until the next customer stop and the corresponding customer time windows in advance, and plans driver activities accordingly (Section 2.8). The numerical results obtained with the mathematical models and the myopic heuristic are analyzed and compared in terms of the run time, lateness and overall schedule duration (Section 2.9).

Chapter 3 focuses on the vehicle refueling subproblem. Again, a fixed sequence of customer locations with time windows is considered. This time, rest periods and breaks of the driver are neglected. Instead, a choice among possible gas stations has to be made and optimal refueling quantities need to be determined. A detailed problem description and outline of the chapter is given in Sections 3.1 and 3.2. Section 3.3 gives an overview of existing literature dealing with vehicle refueling problems. Different graph structures that are proposed in the literature are analyzed in more detail in Section 3.4. In Section 3.5, a mixed integer linear programming (MILP) model for the vehicle refueling problem is presented, referring to the graph structure chosen in the previous section. Section 3.6 shows how a preprocessing of distance, consumption and driving duration data for gas stations can be done and presents a possibility to map gas stations into the main route. The technique proposed is utilized in the following short digression (Section 3.7) that deals with the integration of the proposed MILP model into the vehicle routing problem with time windows (VRPTW) to simultaneously plan vehicle routes and refueling.

The goal of Chapter 4 is to show a possible integration of refueling planning into the models and planning processes described in Chapter 2. After an outline in Section 4.1, it is shown in Section 4.2 how to merge the MILP model from Chapter 3 and the MILP model with optional rules developed in Chapter 2 to simultaneously plan vehicle refueling, customer time windows and driver activities in accordance with Regulation (EC) No 561/2006.
The solution process to solve the resulting multicriteria optimization problem with the help of a commercial optimization solver is described in Section 4.3. The creation of base instances for our numerical experiments is presented in Section 4.4. A heuristic preprocessing procedure which was used to eliminate unattractive gas stations and thus to reduce the problem size and the required computational effort is introduced in Section 4.5. In Section 4.6, the test environment and the different parameter settings are described. An example which shows the evolution of the driver schedule over the optimization steps is presented in Section 4.7. The results of the numerical experiments are then described and analyzed in Section 4.8. In Section 4.9 different elements are described that can be used to develop a heuristic for the combined problem discussed in this chapter.

Chapter 5 concludes with a summary (Section 5.1) and outlines opportunities for future research (Section 5.2).
2. Scheduling of driving times, breaks and rest periods

EU legislation aims at ensuring road safety, adequate working conditions and undistorted competition in the road haulage sector (European Commission (2014)). Regulation (EC) No 561/2006 regulates driving times, breaks and rest periods and Directive 2002/15/EC the working time of drivers in road transport. While Regulation (EC) No 561/2006 is itself obligatory in all member countries, for Directive 2002/15/EC additional national regulations also have to be taken into account.

To control the compliance with the above regulations also referred to as European social legislation, the European Parliament and Council of the European Union (2006a) determine the minimum level of enforcement. The rules provide for checks by authorized inspection officers in the range of 3 to 4% of days worked by drivers. They determine the minimum proportion of checks on the roadside (30%) and at the premises of undertakings (50%). Roadside checks shall be performed at any time and can take place, for example, at service stations or at any other safe locations along highways with the goal to cover the road network sufficiently. In addition to planned checks at premises in accordance with past experience, serious infringements detected are a reason for checks at premises of the corresponding undertakings. To expand and simplify checks, the introduction of the digital tachograph as recording equipment for driver activities was an important step. Its installation is obligatory in all new vehicles that have a mass of more than 3.5 tonnes since 2006. Its application is regulated by Regulation (EU) No 165/2014 (European Parliament and Council of the European Union (2014)).

Transport companies have to organize the work of drivers and instruct them such that they can comply with the social legislation. In particular, Regulation (EC) No 561/2006 stresses the responsibility of all members involved in the transportation process: "Undertakings, consignors, freight forwarders, tour operators, principal contractors, sub contractors and driver employment agencies shall ensure that contractually agreed transport time schedules respect this Regulation" (European Parliament and Council of the European Union (2006b)). For infringements, drivers as well as transport companies may be held responsible and fined severely - and not without reason.

---

According to the European Road Safety Observatory, in 2013, more than 26,000 people died on the roads of the European Union and more than 1.4 million people were injured. More than 15% of the people who died in road accidents in 2013 died in accidents that involved heavy goods vehicles with more than 3.5 tons maximum permissible gross weight (European Road Safety Observatory (2015)). Driver fatigue is involved in 10% to 20% of all road accidents, and several studies suggest that it leads to an increased crash risk. Tired drivers tend to be unfocused, reaction times are increased and the minimum required distance to the vehicle in front may not be respected (SafetyNet (2009)). Sufficient rest periods and breaks help to keep the number of accidents as a consequence of driver fatigue low.

In the transport business, quality of service is an important key factor (European Commission (2014)) that includes punctuality at customer locations. In the EU, converging cost structures will more and more urge transport undertakings to "[...] improve their efficiency and quality of service" (European Commission (2014)). As already mentioned in the introduction (Section 1.1) lateness may be crucial, as it may cause contractual penalties and may lower customer satisfaction that has a significant impact on future requests and thus on the economic viability of a haulage company. The increase in just-in-time management practices even raises the pressure on truck drivers, dispatchers and their transport companies. Besides contractual penalties, a driver that arrives too late at a customer location and thus misses the planned time window may have to wait for several hours until loading or unloading of the vehicle is possible. Additionally, re-planning ties up resources both at the transport company and the customer. An unnecessary early arrival time at a customer location is undesirable as well.

Considering long-haul transport requests, the durations of rest periods and breaks highly influence the overall time needed for fulfillment. They have to be considered when determining arrival times and planning time windows at customer locations and it is important to not only consider their duration but also their time slot in the schedule. For example, daily rest periods and breaks may be split in two parts, but these may not be further divided. At least nine uninterrupted hours are necessary for a daily rest period in which the driver is not allowed to drive or to perform other work. If the schedule only considers resting activities in the form of a fixed proportion of the overall traveling time, deviations from planned arrival times will occur frequently as rest periods and breaks can not be interrupted or split arbitrarily to serve customers and this will lead to the problems described above, especially if narrow time windows are involved.

2.1. Problem description

In this chapter, mathematical models and algorithms that allow to plan driver activities in compliance with Regulation (EC) No 561/2006 for a given sequence of customer locations with one or multiple time windows and other stops to be visited are presented.\textsuperscript{13}

\textsuperscript{13} If a stop corresponds to a customer location with given opening hours, then time windows covering the opening hours reflect this situation.
Distances between consecutive stops and estimated durations for loading, unloading and handling activities are given. Planning may start at the beginning of the week but is also possible during the week. The time management of a driver since the end of the last weekly rest period influences his future activities and is therefore accumulated to the driver status and serves as input for the current (re-)planning phase. This also allows the consideration of online re-planning, i.e adjustments of the original schedule to dynamically respond to unforeseen events.

The goal is to construct a driver schedule that simultaneously considers the choice among possible customer time windows and plans necessary rest periods and breaks to minimize inefficiencies that arise from the distributed decision making of drivers and dispatchers, to increase punctuality and to avoid unnecessary time buffers. In some particular cases, it may not be possible to meet any of the available time windows at a given location. Here, a special feature of our approach is the consideration of "soft" time windows which has not been studied in this context so far. As the dispatcher often has the possibility to negotiate the arrival time of the vehicle with the customer at an early stage, we allow violations of time windows at a penalty. Another reason for soft time windows is that we want to give detailed information even if time windows cannot be met as the resulting schedule gives important information to the dispatcher that is necessary to set up a better schedule. In online re-planning, lateness can be revealed at an early stage such that it is possible to reorganize the schedule or to negotiate arrival times with customers before communication effort and costs increase and further delays or cancellations become unavoidable.

2.2. Outline

For the tasks described in the last section, we developed two MILP models considering a maximum planning horizon of one week. In the following the models presented by Bernhardt et al. (2016) are introduced. In one of the MILP models, the optional rules of Regulation (EC) No 561/2006 are taken into account, in the other one, the optional rules are disabled. The main objective of the two multicriteria optimization problems considered is to minimize the total lateness. With less emphasis, the completion time, i.e. the overall schedule duration until the last customer is serviced and the last stop is reached, is the second optimization criterion. These two criteria form the first set of optimization criteria. Additional criteria that are important for the quality of a solution in practice are taken into account in a second set. As the first set (i.e. lateness and completion time) is considered to be more important than the second one, a lexicographic approach is employed by creating two objective functions, the first one for the first set, the second one for the second set. In each of the objective functions the criteria are provided with different weights. Test instances are derived from real life data. The two MILP models are solved to optimality using a state-of-the-art commercial solver. Afterward, a transformation algorithm translates the solution obtained into a driver schedule that can be implemented in practice. Test results are analyzed.

14 Stops that do not correspond to a customer location are possible. For example, the last stop may be a depot or a rest area that is chosen for the weekly rest period.
The integration of a commercial optimization solver can be costly. A myopic algorithm is also developed that runs without such software as an alternative in order to identify other advantages and disadvantages of the MILP models and a solver.

The following sections are structured as follows. In Section 2.3 the rules implied by Regulation (EC) No 561/2006 are described. Section 2.4 gives a review of the literature that deals with the scheduling of driver activities in compliance with Regulation (EC) No 561/2006. In Section 2.5, the MILP models with and without consideration of the optional rules are introduced, thereby a short digression on modeling techniques used is presented. An algorithm that transforms the model solution into a readable driver schedule is proposed in Section 2.6. Afterwards, in Section 2.7 the test instances are described and different approaches for solving the MILP model with consideration of the optional rules are discussed. Run times for a set of instances based on real-life data are analyzed depending on the number of stops and the number and length of time windows. To examine the influence of the optional rules, the two MILP models are compared with each other considering run times, lateness and completion time. Then, the myopic algorithm is described in Section 2.8. With the test instances presented before, tests are repeated with the short-sighted algorithm and a comparison to the former test results is made in Section 2.9.

2.3. Regulation (EC) No 561/2006

Regulation (EC) No 561/2006 aims at improving working conditions and safety of drivers in road transport laying down provisions concerning maximum driving periods and necessary breaks and rest periods. It applies to the carriage of goods where the maximum permissible mass of the vehicle exceeds 3.5 tonnes or of passengers by vehicles that may carry more than nine persons. It affects all transports exclusively within the European Community or between the European Community, Switzerland and the countries that are part of the Agreement on the European Economic Area. The regulation comprises rules for single drivers and multi-manning.

For all possible driver activities, Regulation (EC) No 561/2006 defines several time periods with specific rules (see Meyer and Kopfer (2008) and Figure 2.1). The rules can be divided in standard rules and optional rules, where adhering to the standard rules suffices to observe the law, while the optional rules allow for more freedom providing alternatives for some of the standard rules.

Before introducing the rules, we give some basic definitions for time periods.

Definitions:

- A rest period is any uninterrupted period of time during which a driver may freely dispose of his or her time. Daily rest periods and weekly rest periods are rest periods.

- A break is a time period exclusively designed for recuperation, during which a driver may not carry out any driving or any other work.
Other work comprises various activities such as loading or unloading, cleaning, technical maintenance, administrative formalities, ensuring safety of the vehicle and its load, etc. Waiting times that are not known in advance are also considered as other work, as the driver cannot dispose freely of his time and is required to be at his workstation.

Driving time is the duration of driving and includes all activities related to driving, even when the vehicle is temporarily not in motion, for example, when waiting at traffic lights or in a traffic jam.

A week means the period of time between 00:00 on Monday and 24:00 on Sunday.

Standard rules:

1. A break has a duration of at least 45 minutes.
2. A daily rest period has a duration of at least 11 hours.
3. A weekly rest period has a duration of at least 45 hours.
4. The accumulated driving time between a rest period or a break and another rest period or break is restricted to a maximum of 4.5 hours.
5. The daily driving time, i.e. the total accumulated driving time between the end of one rest period and the beginning of the following rest period, is restricted to a maximum of 9 hours.
6. Within each period of 24 hours after the end of the previous rest period, a driver must have taken a new daily rest period. This means that a driver must take a daily rest period at most 13 hours after he has completed the previous daily or weekly rest period.
7. A weekly rest period must start no later than 144 hours (six 24-hour periods) after the end of the previous weekly rest period.
8. The weekly driving time, i.e. the total accumulated driving time between 00:00 on Monday and 24:00 on Sunday, is not allowed to exceed 56 hours.
9. The total accumulated driving time during any two consecutive weeks must not exceed 90 hours.

10. In any two consecutive weeks a driver has to take at least two weekly rest periods. Weekly rest periods that fall in two weeks may be counted in either week, but not in both.

11. The maximum weekly working time is not allowed to exceed 60 hours.

12. Over four months, the average weekly working time may not exceed 48 hours.


Optional rules:

1. A break may be taken in two parts, the first part having a duration of at least 15 minutes followed by the second one of at least 30 minutes.

2. A regular daily rest period may also be split into two parts with the first one having a duration of at least 3 hours and the second one having a duration of at least 9 hours.

3. The duration of a daily rest period may be reduced to at least 9 hours at most 3 times between two weekly rest periods. In the case that the next daily rest period is planned to be a reduced one, that rest period has to start at most 15 hours after the completion of the previous daily or weekly rest period.

4. The daily driving time may be extended to at most 10 hours not more than twice during a week, where a week means the period of time between 00:00 on Monday and 24:00 on Sunday.

5. In two consecutive weeks, the duration of one of the two weekly rest periods may be reduced to 24 hours. However, the reduction has to be compensated by an equivalent period of rest taken at a time before the end of the third week following the week containing the reduced weekly rest period. Any rest taken as compensation for a reduced weekly rest period must be attached to another rest period of at least 9 hours.

Deviating from the standard rules 2 and 6 and making the optional rules 2 and 3 needless, a driver engaged in multi-manning must have taken a new daily rest period with a duration of at least 9 hours within 30 hours of the end of a daily or weekly rest period. In the following, we will focus on the single truck driver case.

Furthermore, additional rules apply for special cases as for example for accompanying vehicles transported by ferry or train, the traveling to a location to take charge of a vehicle, for driving times not falling in the scope of Regulation (EC) No 561/2006 or special vehicles.

For transport between EU and non-EU countries (third countries) the AETR agreement (see United Nations Economic Commission for Europe (2006)) regulates the work of drivers engaged in international transport and covers 49 contracting parties including all EU
Member States (European Commission (2016)). The implementation into national law is mandatory. Its provisions concerning rest periods and breaks are similar to Regulation (EC) No 561/2006, with the difference that drivers engaged in multi-manning are exempted from standard rule 10 and optional rule 5.

2.4. Literature review

In the past few years, research on including regulations concerning rest periods and breaks in operational transportation planning has attracted increasing attention especially in combination with vehicle routing. The following review concentrates on literature that deals with Regulation (EC) No 561/2006 and the single truck driver case.


Goel and Gruhn (2006) and Goel (2009) present a large neighborhood search algorithm to solve the combined problem of vehicle routing with time windows and scheduling driver activities in consideration of Regulation (EC) No 561/2006 for single-manned vehicles and for a planning horizon of one week. For the scheduling sub-problem, Goel (2009) introduces a naive labeling algorithm and a multilabeling algorithm to plan driver activities according to a subset of the regulations. The naive method only schedules breaks and rest periods if the accumulated driving time is exhausted or enough idle time before the lower boundary of the time window at a customer is available, whereas the multilabeling method allows for earlier breaks and rest periods if the corresponding accumulated driving time reaches its maximum to be able to reach narrow time windows. Optional regulations are excluded and the regulation that there has to be a daily rest period within 24 hours after the end of the previous daily rest period (standard rule 6) is only considered in a post-processing step by Goel (2009), who presents a repair method. New benchmark instances based on the well-known instances by Solomon (1987) for the vehicle routing problem with time windows (VRPTW) are set up.

Derigs et al. (2011) extend the work of Bartodziej et al. (2009) and propose a checking procedure for route feasibility that is motivated by Goel (2009). Splitting of daily rest periods and breaks, reduced daily rest periods and the possibility to extend the daily driving time are considered in the procedure.

In addition to Goel (2009), Kok et al. (2010) consider all optional rules and also Directive 2002/15/EC that supplements the rules laid down in Regulation (EC) No 561/2006 with additional restrictions for the working time of persons engaged in road transportation, extending the naive labeling method presented by Goel (2009). They incorporate the EC social legislation in a restricted dynamic programming algorithm by adding state dimensions. Breaks are scheduled in constant time by using a constructive solution method with a break scheduling algorithm that decides locally when breaks have to be scheduled. Their test results show significant improvements concerning the number of vehicles needed and
the distance traveled with less computation time than Goel (2009). In particular, Kok et al. (2010) show that including the optional rules of Regulation (EC) No 561/2006 allows for additional flexibility and can reduce costs significantly. However, similar to Goel (2009), they cannot guarantee to find a feasible driver schedule for a route, even if one exists.

Given a sequence of locations to be visited within specified time windows, Goel (2010) presents a method for scheduling driving and working hours of truck drivers with respect to Regulation (EC) No 561/2006. The feature of this approach is the guarantee to find a schedule complying with the regulation if such a schedule exists. Goel (2010) introduces conditions for pseudo-feasibility which relax the conditions for feasibility and gives dominance criteria, thus reducing the number of partial schedules that have to be explored. However, he neglects the possibility of extending daily driving times and of reducing the duration of daily rest periods.

Drexl and Prescott-Gagnon (2010) describe an exact algorithm and two heuristic approaches for considering European rules for rest periods and breaks in shortest path problems with resource constraints. The proposed labeling algorithms are based on the idea of so-called resource extension functions to expand labels to plan rest periods and breaks.

Prescott-Gagnon et al. (2010) present a large neighborhood search method for the vehicle routing problem with time windows and driver regulations. In this method, the neighborhoods are explored using a column generation heuristic that relies on a tabu search algorithm. The tabu search heuristic allows two possible route modifications, the deletion or the insertion of a customer. While route feasibility is maintained if a customer is removed, the insertion of a new customer requires a feasibility check. Therefore, Prescott-Gagnon et al. (2010) develop a heuristic procedure that uses labels with resource components and resource extension functions. Numerical results show that significant improvements can be achieved compared to the procedures proposed by Kok et al. (2010) and Goel (2009).

Kopfer et al. (2007) analyze the influence of the European social legislation on vehicle routing and scheduling, and are the first ones to propose a mathematical formulation. Later, Kopfer and Meyer (2009), Kopfer and Meyer (2010), and Meyer (2011) develop a MILP formulation to map Regulation (EC) No 561/2006. Kopfer and Meyer (2009) use a position-based formulation of the traveling salesman problem with time windows (TSPTW) to integrate the rules of the regulation. Kopfer and Meyer (2010) and Meyer (2011) continue with an extension for the VRPTW, also including Directive 2002/15/EC. They solve randomly generated test instances with CPLEX. Unfortunately, if long distances have to be traveled between two consecutive customers, as it is often the case in international transports, the model is not applicable, as it is presumed that driving between two customer locations will not require more than one daily rest period. The constraints that are used to model the restriction that there has to be a daily rest period in each 24 hours time interval only demand a number of daily rest periods proportional to the overall schedule duration and thus are problematic if daily rest periods are scheduled earlier than required. The solution specifies the number of breaks and rest periods between two consecutive customers. The transformation algorithm that is necessary to determine a driver schedule that includes the exact timing and the sequencing of rest periods and breaks is not described in detail.
Kok et al. (2011b) present a model for departure time optimization as a post-processing step of the VRPTW that incorporates the European driving hours regulations and time-dependent travel times. They assume that breaks have to be taken at customer locations. Here, a planning horizon of a single working day is considered and computational results are discussed. Additional constraints to model a planning horizon of multiple days and the possibility to take breaks at parking lots are proposed, but the option of splitting daily rest periods into two parts is neglected. No test results for the extended model are presented.

Kok et al. (2011a) propose a restricted dynamic programming heuristic for the VRPTW with time-dependent travel times and EC social legislation that is restricted to the planning horizon of one day.

Goel (2012) presents a mixed integer programming formulation for a variant of the truck driver scheduling problem in which drivers only may rest at customer locations and rest areas and shows how to model rules commonly found in different hours of service regulations. Rest areas are modeled as dummy locations with zero duration for loading and unloading and unbounded time windows. A dynamic programming approach is proposed that is able to solve the problem efficiently and it is shown how additional rules like the optional rules of Regulation (EC) No 561/2006 for splitting breaks and daily rest periods can be incorporated. However, it is assumed that rest areas are roadside, as no detours are considered. Test instances are randomly generated for a planning horizon of one week that ends on Friday, rest areas are randomly distributed, and up to 4 time windows per customer location with time windows from 6:00 to 20:00 or from 6:00 to 12:00 and 14:00 to 20:00 on one or two days are considered. It is shown that the availability of suitable rest areas has a significant impact on the number of instances for which feasible schedules could be found.

Goel and Vidal (2014) use a hybrid genetic search with advanced diversity control for solving the combined vehicle routing and truck driver scheduling problem. Truck driver scheduling is done for route evaluations with adjustments of the forward labeling algorithms developed for the rules applied in different countries and areas, among these the rules of the European Union. Considering Directive 2002/15/EC, the authors include the same set of rules as Prescott-Gagnon et al. (2010). They consider multiple time windows and allow penalized lateness with respect to the time window constraints. However, lateness is only allowed to facilitate transition between structurally different solutions during the search and there, any voluntary increase in lateness at a customer location for the purpose of reducing lateness at subsequent customers is forbidden. Furthermore, Goel and Vidal (2014) give an international comparison of the economic impact of different hours of service regulations.

2.5. Mathematical formulation

We start by describing the initial situation. A dispatcher has assigned transport requests to a vehicle. Each request consists of a pickup and a corresponding delivery location (customer locations). The dispatcher has determined a sequence in which all stops associated with customer requests have to be visited (see for example Figure 2.2).
Each customer location has at least one time window, i.e. a time interval in which the loading and/or unloading of goods should start. In contrast to Kopper and Meyer (2010), we consider multiple customer time windows, as in reality, a dispatcher often has the possibility to choose among a set of time windows proposed by a customer. Loading or unloading does not have to be finished before the end of the chosen time window. This is just a modeling decision. If in practice the loading and unloading tasks should take place within the time window, the estimated duration is subtracted from the end of the interval to narrow the time window accordingly. Note that a time window that lies beyond the maximum time interval of $6 \cdot 24 \text{h} = 144 \text{h}$, that starts with the end of the last weekly rest period, will avoid finding a feasible schedule, as standard rule 7 cannot be met. This can be checked in advance.

Modeling time windows as hard constraints like it is done in the literature (recall Section 2.4) has one major drawback. If no schedule can be found because time windows cannot be met, the dispatcher only gets the information that there is "no solution". We use soft time windows, that means that we penalize lateness but do not prohibit it (see objective function (2.5.184) and constraints (2.5.77)) which has the advantage that still a schedule may be returned even if time windows cannot be met. Unavoidable lateness is revealed
and the dispatcher is given a schedule that may help him to re-plan or re-negotiate time windows with the customers.

Only the constraints that take care of the maximum time interval between two consecutive weekly rest periods may avoid that a schedule can be found (see constraints (2.5.78) on page 58). In such a case, constraints (2.5.78) can be removed and the MILP model can be re-solved to obtain a solution that is not feasible in practice, but may be an important information for the dispatcher.

The current vehicle position or starting point of the tour and all customer locations are represented by vertices, where vertex 0 denotes the vehicle position at the start of the planning horizon (depicted as a green point in Figure 2.2) and vertices 1, ..., r − 2 denote the customer locations (customer vertices) that are numbered in the order they have to be visited (see Figure 2.3). As drivers do not necessarily return to a specific location like a central depot when doing long-haul trips, the last vertex represents the last known customer location to be visited or, if known in advance, the location where the driver will take the weekly rest period (red point in Figure 2.2). That may be a rest area near to the route to the subsequent customer location or near to a customer location if loading or unloading the vehicle should start after the weekly rest period.

Figure 2.3.: Vertices

In each customer vertex\(^\text{15}\), the driver may wait, take a break or a daily rest period before loading or unloading tasks. If the daily working time still left when entering a vertex does not suffice to also carry out the loading/unloading tasks, a daily rest period is scheduled. An interruption by daily rest periods is thus avoided. Generally, rest periods and breaks may be taken to reduce or eliminate the waiting time. This also includes a first part of a splitted break or rest period if this helps to reduce the time needed for resting activities on subsequent arcs and in vertices. Figure 2.4 summarizes the activities that can be performed at a vertex.

Possible activities on an arc comprise driving, taking breaks and daily rest periods (see Figure 2.5). Note that the first part of a split break may only be scheduled in vertices to compensate waiting time, whereas the first part of a daily rest period may also be scheduled as last resting activity on an arc instead of a break. The first part of a daily rest period, similar to a break, resets the 4.5 hours driving contingent. If the first part of a daily rest period is taken, an additional break to be able to use up the complete daily driving time is only necessary if a driving time extension is considered. In some cases, it may

\(^{15}\) For ease of reference, and to connect modeling aspects to the real life application, the term "vertex" is used in association with customer locations. Analogously, the term "arc" is used, when talking about the route between customer locations and activities on this route.
be advantageous to take a first partial daily rest period instead of a break as last resting activity on an arc (for more details see Section 2.5.4 et seq.). Reduced daily rest periods and breaks needed for a driving time extension may be scheduled on arcs and vertices as well.

![Driver time management activities in a vertex](image)

**Figure 2.4.:** Driver time management activities in a vertex

![Driver time management activities on an arc](image)

**Figure 2.5.:** Driver time management activities on an arc

Similar to the literature presented in Section 2.4, we consider a planning horizon of at most one week that ends with the start of the next weekly rest period and concentrate on single manned vehicles. As the overall driving and working time needed is already given with the input data, a check can be made if the weekly driving (see standard rule 8) and working time (see standard rule 11) allowed suffice to fulfill all scheduled customer requests. The biweekly driving time (standard rule 9) can be considered by adding the driving time of the last week to the planned driving time for the current week and testing whether the result is less than 90 hours. Similarly, compliance with the average weekly working time over four months (standard rule 12) can be checked. Standard rule 10 and optional rule 5 are not considered by the MILP models and compliance with them has to be ensured externally when concatenating two planning horizons with the first one ending before the start of a weekly rest period and the other one starting after the end of the weekly rest period.

---

Note that waiting time known in advance is not considered as working time.
2.5. Mathematical formulation

Usually, depending on the countries visited, it is advantageous to include Sundays in weekly rest periods because of the ban on movement of goods vehicles on Sundays in many countries of the European Union. Therefore, we assume that the time between two weekly rest periods does not range across more than one week, i.e. the time between 00:00 on Monday and 24:00 on Saturday.

Drivers may help when loading and unloading the vehicle but this does not have to be the case. At least, they have to perform handling activities. We assume that the drivers have to be present when loading or unloading takes place and consequently do not interpret this time as a break and do not allow it to be used as part of a daily rest period.

In our models, no use is made of constraints which give a lower bound on the number of rest periods and breaks needed for every number of consecutive arcs as it is done in Kopfer et al. (2007). Instead, status variables are introduced which map the driver status when entering and leaving a vertex, and thus link activities on different arcs and in different vertices. These status variables reflect, for example, the amount of time left for different activities without taking a break or daily rest period when entering or leaving a vertex:

- \( E_{dt}^i, L_{dt}^i \): driving time left until the next break or daily rest period when entering vertex \( i \) or leaving, respectively.
- \( E_{ddt}^i, L_{ddt}^i \): daily driving time left until the next daily rest period when entering vertex \( i \) or leaving, respectively.
- \( E_t^i, L_t^i \): overall time left until the next daily rest period when entering vertex \( i \) or leaving, respectively.

Other status variables keep track of driving time extensions and reduced daily rest periods that were previously scheduled. Moreover, if a break or a daily rest period is split into two parts, status variables keep track if a first part has already been taken.

Dependencies between activities on arcs and the status variables are shown in Figure 2.5. Figure 2.4 depicts the dependencies when entering and leaving a vertex.

The models allow setting start values for the status variables. Thus, online re-planning during the week with updated tour information including remaining stops in the original schedule and new ones can take place. If there are deviations from the original plan, e.g. an increased travel time due to traffic congestion, and time windows cannot be met or the time left does not suffice to visit all locations planned, with online re-planning dispatchers get the possibility to recognize such deviations early and to re-negotiate time windows or remove stops from the vehicle schedule.

The models explore the possibility of taking a daily rest period earlier than after 9h of driving even on the path between two customers. This is advantageous if the time for a break can be saved thus preventing a time window from being violated. This is illustrated in Figure 2.6.\(^{17}\) The first schedule leads to a lateness of 15 minutes, whereas in the second schedule with an early daily rest period the time window at customer \( i + 1 \) is met.

\(^{17}\) The symbols used are depicted in Figure 1.7 on page 8
2. Scheduling of driving times, breaks and rest periods

Figure 2.6.: Partial schedule with vs. without early daily rest period

In a vertex, it is always possible to schedule a break or a daily rest period to compensate waiting time. Figure 2.7 shows an example, where taking a break in a vertex is necessary to be on time at the subsequent customer location.

Figure 2.7.: Partial schedule with vs. without a break in vertex \( i \)

With regard to an application in international freight transportation, where often long distances have to be traveled, we wish to exactly determine the time intervals for each planned driver activity. Therefore, an algorithm was developed to transform the model solution into a detailed driver schedule that gives information to both the driver and the dispatcher. This algorithm can be found in Section 2.6.

The remainder of this section is structured as follows. After a short digression on modeling
techniques, model parameters and variables will be introduced. Then, the MILP model with consideration of the optional rules is presented, whereby the description of the objective functions and the constraints is split into several subsections. How to switch off the optional rules is described at the end of this section. Note that all durations and lengths of time intervals are expressed in minutes.

2.5.1. Modeling techniques - A digression on modeling logical conditions with binary variables

For modeling different kinds of decisions such as taking a daily rest period or making use of one of the optional rules, and modeling different kinds of driver states that will be introduced later, binary (zero-one) decision variables will be used. For example, \( \alpha_{i}^{\text{rest}} \) is a binary variable that takes the value 1 if a daily rest period should be taken at customer location \( i \), and 0 otherwise.

As described in the previous section, allowed driver activities in a vertex depend on former decisions about activities on the previous arc. The latter in turn depend on activities scheduled for the previous vertex, and so on. To express these dependencies in the model, status variables link the activities in successive arcs and vertices. Activities are modeled by indicator and integer variables, whereas continuous variables reflect their duration. The goal here is to link the continuous status variables to indicator variables for activities.

Depending on the relation of several driver status variables to other driver status variables for entering or leaving a vertex \( i \), the values of driver status variables for leaving vertex \( i \) or entering vertex \( i + 1 \), respectively, are determined. A good example is the daily driving time left \( L_{i}^{\text{ddt}} \) when leaving vertex \( i \) (see pages 76 et seq.). Here, indicator variables are used to determine if other status variables set up lower and upper bounds on \( L_{i}^{\text{ddt}} \).

Generally, when formulating conditions, we can differentiate between the following two possibilities that reflect the direction of dependency:

- the value of a variable (continuous or integer) is derived from the value of one or several binary variables.
- the value of a binary variable is derived from other variables (binary, integer or continuous variables) and their relation to each other.

We will now go into further detail for the possibilities mentioned above. A more general description of transforming logical conditions into linear constraints by the usage of binary variables can be found in Williams (2013).

Binary variables inducing the value of other variables

We start by addressing the simple case that from the binary variable \( \delta \) being equal to 1 it follows that the variable \( x \) is equal to \( y \). \( y \) itself may be a linear expression. That means,
we wish to state that
\[ \delta = 1 \Rightarrow x = y \] (a)

Therefore, we first reformulate (a) by (b):
\[
\begin{align*}
\delta = 1 & \Rightarrow x \leq y & \text{and} \\
\delta = 1 & \Rightarrow x \geq y
\end{align*}
\] (b)

Now, the condition can be induced by two linear constraints:
\[
\begin{align*}
x \leq y + M_1 (1 - \delta) & \quad \text{(c)} \\
x \geq y - M_2 (1 - \delta) & \quad \text{(d)}
\end{align*}
\]

\( M_1 \) has to be chosen as an upper bound on \( x - y \) such that constraint (c) becomes redundant in case that \( \delta = 0 \). Similarly, \( M_2 \) has to be chosen as a lower bound on \( x - y \). It is important not to choose \( M_1 \) and \( M_2 \) too small as this will introduce constraints on the difference between \( x \) and \( y \) that we do not want to model. In turn, taking very large values for \( M_1 \) and \( M_2 \) may result in numerical problems. Therefore, Williams (2013) recommends to choose upper bounds as small as possible and lower bounds as large as possible.

If more than one binary variable needs to take on the value 0 or 1 in order for \( x \) to be equal to \( y \), several "big-M terms" have to be added. As an example, we address the combination of two binary decision variables.

First of all, let us consider the case that the given values of two variables (example: \( \delta_1 = 1 \) and \( \delta_2 = 0 \)) induce \( x = y \). The logical condition
\[
(\delta_1 = 1 \wedge \delta_2 = 0) \Rightarrow x = y
\] (e)

can be split into
\[
\begin{align*}
(\delta_1 = 1 \wedge \delta_2 = 0) & \Rightarrow x \leq y \\
(\delta_1 = 1 \wedge \delta_2 = 0) & \Rightarrow x \geq y
\end{align*}
\] (e1) (e2)

This now can be modeled by the following linear constraints
\[
\begin{align*}
x \leq y + M_1 (1 - \delta_1) + M_1 \delta_2 & \quad \text{(e3)} \\
x \geq y - M_2 (1 - \delta_1) - M_2 \delta_2 & \quad \text{(e4)}
\end{align*}
\]

with \( M_1 \) and \( M_2 \) being appropriate upper and lower bounds, respectively.

If \( y \) itself is already an upper bound or lower bound on \( x \), no additional constraint with a big-M term needs to be added to the model. If additionally to (e), it is known that already
2.5. Mathematical formulation

$\delta_1 = 1$ induces $x \leq y$, condition (e3) is substituted by

$$x \leq y + M_1 (1 - \delta_1) \quad \text{(e3')}
$$

Analogously, this can be done if $\delta_1 = 1$ induces $x \geq y$ or if $\delta_2 = 0$ induces $x \leq y$ or $x \geq y$.

The "exclusive or" in $(\delta_1 = 1 \lor \delta_2 = 1) \Rightarrow x = y$ is equivalent to the combination of

$$(\delta_1 = 1 \land \delta_2 = 0) \Rightarrow x = y \quad \text{and}
$$

$$(\delta_1 = 0 \land \delta_2 = 1) \Rightarrow x = y
$$

and can therefore be expressed by using the scheme described above. In case that

$$\delta_1 + \delta_2 \leq 1
$$

the following two additional inequalities suffice:

$$x \leq y + M_1 (1 - \delta_1 - \delta_2)
$$

$$x \geq y - M_2 (1 - \delta_1 - \delta_2)
$$

This will be used when deriving the daily driving time left $L_{i}^{d_{t}}$ when leaving vertex $i$ on pages 76 et seq.

Binary variables derived from other variables

Now, the opposite case is considered. The value of binary variables is determined depending on the value of other variables, being either integer or continuous. A dependency we will often need later (see for example pages 76 for $\lambda_{1}^{i}$ to $\lambda_{3}^{i}$) is the following one: in case $x$ is greater than $y$, the binary variable $\delta$ should be equal to 1. If $x$ is less than $y$, $\delta$ is set to be equal to 0. If $x = y$, we do not care about the value of $\delta$.\(^\text{18}\)

This means that we wish to state that

$$x > y \Rightarrow \delta = 1
$$

$$x < y \Rightarrow \delta = 0
$$

\(^{18}\)In principle, in the cases we have to consider, we have to model a piecewise linear continuous function depending on $x$. In the simplest case, this function $f$ is composed of two parts with

$$f(x) = \begin{cases} 
  g(x) & \text{if } x \leq y \\
  h(x) & \text{if } x > y
\end{cases}
$$

For $x = y$ it does not matter which function is evaluated ($g$ or $h$) to determine $f(x)$, as $g(y) = h(y)$, because $f$ is continuous.
This can be done by using the following constraints:

\[ M_1 \delta \geq x - y \]  
\[ M_2 (\delta - 1) \leq x - y \]

where (f1) induces \( x > y \Rightarrow \delta = 1 \) and (f2) induces \( x < y \Rightarrow \delta = 0 \). \( M_1 \) has to be an upper bound on \( x - y \) whereas \(-M_2\) has to be a lower bound on \( x - y \). This procedure is used for determining the value of the binary variables \( \lambda^k_i, k = 1, \ldots, 6 \), which depend on several continuous status variables and activities in vertex \( i \) and arc \((i, i + 1)\) (see for example pages 76 et seq. for \( \lambda^1_i, \lambda^2_i \) and \( \lambda^3_i \)).

Sometimes, we need to express that a binary variable \( \delta \) is equal to one if and only if the integer variable \( x \in \mathbb{N}_0 \) is greater than zero, and \( \delta \) is equal to zero, otherwise.

\[ x > 0 \Leftrightarrow \delta = 1 \]

This is ensured by

\[ M \delta \geq x \]  
\[ \delta \leq x \]

where \( M \) is an upper bound on \( x \). This is required, for example, for determining the binary decision variable \( \alpha^r_{\text{rest}}(i, i+1) \) which depends on the integer variable \( \Delta^r_{\text{rest}}(i, i+1) \) and vice versa on page 59.

The modeling techniques introduced will be used to develop the MILP models. But before, parameters and variables are described in the following two sections.

### 2.5.2. Parameters of the model

- \( r \in \mathbb{N} \) Total number of vertices. The vertices are numbered from 0 to \( r - 1 \) according to the sequence of customer locations to be visited. The first vertex (0) represents the position of the vehicle at the beginning of the planning horizon, the last vertex (\( r - 1 \)) represents the last location (see Figure 2.3)
- \( \bar{\Delta}^{\text{drive}}_{(i, i+1)} \in \mathbb{N}_0 \) Driving time in minutes needed to travel from \( i \) to \( i + 1 \), \( i = 0, \ldots, r - 2 \)
- \( \bar{\Delta}^{\text{service}}_i \in \mathbb{N}_0 \) Time needed for loading and/or unloading the vehicle at vertex \( i \), \( i = 0, \ldots, r - 1 \), in minutes, \( \bar{\Delta}^{\text{service}}_0 = 0 \)
- \( \text{noTW}_i \in \mathbb{N} \) Number of time windows at customer location \( i \), \( i = 1, \ldots, r - 1 \)
- \( \text{TW}_{i z}^{\text{begin}} \in \mathbb{N}_0 \) Lower limit of the time window \( z \) at vertex \( i \), \( i = 1, \ldots, r - 1 \), \( z = 0, \ldots, \text{noTW}_i - 1 \) in minutes counted from start time 0
2.5. Mathematical formulation

\( T_\text{end} \in \mathbb{N}_0 \) Upper limit of the time window \( z \) at vertex \( i, i = 1, \ldots, r - 1, \\ z = 0, \ldots, noTW_i - 1 \) in minutes counted from start time 0

\( udt \in \mathbb{N}_0 \) Driving time since the last daily rest period or break at the beginning of the planning horizon in minutes

\( ddt \in \mathbb{N}_0 \) Cumulated daily driving time since the end of the last daily rest period at the beginning of the planning horizon in minutes

\( ptr \in \mathbb{N}_0 \) Elapsed time since the end of the last daily rest period at the beginning of the planning horizon in minutes

\( ptwr \in \mathbb{N}_0 \) Elapsed time since the end of the last weekly rest period at the beginning of the planning horizon in minutes

\( urt \in \mathbb{N}_0 \) If a daily rest period takes place at start time, this parameter expresses its duration since its beginning in minutes

\( ubt \in \mathbb{N}_0 \) If a break takes place at start time, this parameter expresses its duration since its beginning in minutes

\( dte \in \{0, 1\} \) Is equal to 1 if a driving time extension is currently used when the planning horizon begins, 0 otherwise

\( hpb \in \{0, 1\} \) Is equal to 1 if the first part of a break with a duration of at least 15 minutes has already been taken before the beginning of the planning horizon, 0 otherwise

\( hpr \in \{0, 1\} \) Is equal to 1 if the first part of a daily rest period with a duration of at least 3 hours has already been taken before the beginning of the planning horizon, 0 otherwise

\( no\text{Red} \in \{0, 1, 2, 3\} \) The number of reduced daily rest periods that have already been taken in the current week

\( no\text{Ext} \in \{0, 1, 2\} \) The number of extended daily driving times that have already been taken in the current week

2.5.3. Variables

Variables needed to define the objective function \((2.5.184)\)

\( start_i \in \mathbb{R}_0^+ \) Start of service time in vertex \( i, i = 1, \ldots, r - 1, \)

Start of driving (after potential break or rest) for \( i = 0 \)

\( \Delta_{\text{late}} \in \mathbb{R}_0^+ \) Lateness in vertex \( i, i = 1, \ldots, r - 1 \)

Variables that indicate which time window is chosen at customer location \( i \)

\[
\begin{align*}
tw_{iz} &= \begin{cases} 1 & \text{if time window } z \text{ is chosen at destination } i \\ 0 & \text{otherwise} \end{cases} \\
i &= 1, \ldots, r - 1, z = 0, \ldots, noTW_i - 1
\end{align*}
\]
The following set comprises the continuous status variables for each vertex $i$.

- $E_{dt}^{i}$: Driving time left until the next break or daily rest period when entering vertex $i$, $i = 0, \ldots, r - 1$ in minutes
  
  $0 \leq E_{dt}^{i} \leq 270$

- $E_{ddt}^{i}$: Driving time left until the next daily rest period when entering vertex $i$
  
  $i = 0, \ldots, r - 1$ in minutes

  $0 \leq E_{ddt}^{i} \leq 540$

- $E_{t}^{i}$: Time left until the next daily rest period when entering vertex $i$
  
  $i = 0, \ldots, r - 1$ in minutes

  $0 \leq E_{t}^{i} \leq 900$

- $L_{dt}^{i}$: Driving time left until the next break or daily rest period when leaving vertex $i$, $i = 0, \ldots, r - 1$ in minutes
  
  $0 \leq L_{dt}^{i} \leq 270$

- $L_{ddt}^{i}$: Driving time left until next daily rest period when leaving vertex $i$
  
  $i = 0, \ldots, r - 1$ in minutes

  $0 \leq L_{ddt}^{i} \leq 540$

- $L_{t}^{i}$: Time left until the next daily rest period when leaving vertex $i$
  
  $i = 0, \ldots, r - 1$ in minutes

  $0 \leq L_{t}^{i} \leq 900$

The following variables indicate for each arc $(i, i+1)$ if a daily rest period is taken, the number of daily rests periods and their cumulative duration.

- $\alpha_{rest}^{(i,i+1)} = \begin{cases} 
1 & \text{if at least one daily rest period is taken on arc } (i, i+1) \\
0 & \text{otherwise} 
\end{cases}$

  $i = 0, \ldots, r - 2$

- $A_{rest}^{(i,i+1)} \in \mathbb{N}_0$ The number of daily rest periods taken on arc $(i, i+1)$,
  
  $i = 0, \ldots, r - 2$

- $\Delta_{rest}^{(i,i+1)} \in \mathbb{R}_0^+$ The cumulative duration of all daily rest periods on arc $(i, i+1)$,
  
  $i = 0, \ldots, r - 2$

Regarding daily rest periods at vertices, the following variables indicate if a daily rest period is taken and its duration.

- $\alpha_{rest}^{i} = \begin{cases} 
1 & \text{if a daily rest period is taken in vertex } i \\
0 & \text{otherwise} 
\end{cases}$

  $i = 0, \ldots, r - 1$
The duration of a daily rest period in vertex $i$, $i = 0, \ldots, r - 1$.

The next set of variables are needed to determine if breaks are taken on arc $(i, i+1)$ and their number.

$$
\alpha_{\text{break}}^{i+1} = \begin{cases} 
1 & \text{if at least one break is taken on arc } (i, i+1) \\
0 & \text{otherwise}
\end{cases}
$$

$i = 0, \ldots, r - 2$

The number of breaks taken on arc $(i, i+1)$, $i = 0, \ldots, r - 2$.

The following variables indicate if breaks are taken in vertices.

$$
\alpha_i^{\text{break}} = \begin{cases} 
1 & \text{if a break is taken in vertex } i \\
0 & \text{otherwise}
\end{cases}
$$

$i = 0, \ldots, r - 1$

Each variable $\Delta_i^{\text{wait}}$ gives the waiting time in vertex $i$:

$$
\Delta_i^{\text{wait}} \in \mathbb{R}_0^+ \quad \text{Waiting time in vertex } i, i = 0, \ldots, r - 1
$$

The next variables specify if an early daily rest period is taken on an arc, meaning that the daily driving time is not completely used up.

$$
\mu_{\text{earlydr}}^{i+1} = \begin{cases} 
1 & \text{if a break is replaced by a daily rest period on arc } (i, i+1) \\
0 & \text{otherwise}
\end{cases}
$$

$i = 0, \ldots, r - 2$

$$
\mu_{\text{earlydr}}^{i+1} = \begin{cases} 
1 & \text{if a break is replaced by a daily rest period on arc } (i, i+1) \\
0 & \text{otherwise}
\end{cases}
$$

$i = 0, \ldots, r - 2$
When arriving in vertex $i$, in case a daily rest period was taken on arc $(i-1, i)$, the following variable indicates if a break was taken since the last daily rest period.

$$e_{bt}^i = \begin{cases} 
1 & \text{if the last rest activity on the preceding arc } (i-1, i) \text{ was a break} \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$

The next variables indicate if a break is still necessary to completely use up the daily driving time left when leaving vertex $i$.

$$l_{bn}^i = \begin{cases} 
1 & \text{if a break would be necessary to completely exploit} \\
0 & \text{the daily driving time left when leaving vertex } i \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$

The following variables are needed to model the optional rules.

$$\alpha_{pbreak}^i = \begin{cases} 
1 & \text{if the first part of a break is taken in vertex } i \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$

$$\mu_{upbreak}^{(i, i+1)} = \begin{cases} 
1 & \text{if the second part of a break is taken on arc } (i, i+1) \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 2$

$$\mu_{upbreak}^{i} = \begin{cases} 
1 & \text{if the second part of a break is taken in vertex } i \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$

$$l_{pbreak}^i = \begin{cases} 
1 & \text{if when leaving vertex } i \text{ a partial break of 15 minutes was taken} \\
0 & \text{since the last rest period} \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$
\[ \alpha_i^{\text{prest}} = \begin{cases} 
1 & \text{if the first part of a daily rest period is taken in vertex } i \\
0 & \text{otherwise}
\end{cases} \\
i = 0, \ldots, r - 1
\]

\[ p_i^{\text{prest}} = \begin{cases} 
1 & \text{if when leaving vertex } i \text{ a partial rest period of } 3 \text{ h was taken} \\
0 & \text{since the last rest period}
\end{cases} \\
i = 0, \ldots, r - 1
\]

\[ \mu_i^{\text{prest}} = \begin{cases} 
1 & \text{if the last break on arc } (i - 1, i) \text{ is substituted by a} \\
0 & \text{first partial rest}
\end{cases} \\
i = 1, \ldots, r - 1
\]

\[ \mu_i^{\text{dredrest}} = \begin{cases} 
1 & \text{if in vertex } i \text{ the decision is made that the next} \\
0 & \text{daily rest period after leaving vertex } i \text{ will be a reduced one}
\end{cases} \\
i = 0, \ldots, r - 1
\]

\[ \mu^{\text{redrest}}_{(i, i+1)} \in \{0, 1, 2, 3\} \quad \text{The number of reduced daily rest periods taken} \]
\[ \text{on arc } (i, i + 1), \; i = 0, \ldots, r - 2
\]

\[ \mu_i^{\text{dredrest}} = \begin{cases} 
1 & \text{if a reduced daily rest period is taken in vertex } i \\
0 & \text{otherwise}
\end{cases} \\
i = 0, \ldots, r - 1
\]

\[ l_i^{\text{dredrest}} = \begin{cases} 
1 & \text{if the next daily rest period is a reduced one and is taken} \\
0 & \text{after leaving vertex } i
\end{cases} \\
i = 0, \ldots, r - 1
\]
2. Scheduling of driving times, breaks and rest periods

\[ \mu_{(i,i+1)}^{extd1} = \begin{cases} 1 & \text{if a driving time extension is used on arc } (i, i+1) \text{ before the first daily rest period} \\ 0 & \text{otherwise} \end{cases} \]

\[ i = 0, \ldots, r - 2 \]

\[ \mu_{(i,i+1)}^{extd2} \in \{0, 1, 2\} \quad \text{The number of driving time extensions used on arc } (i, i+1) \text{ between the first and the last daily rest period, } i = 0, \ldots, r - 2 \]

\[ \mu_{(i,i+1)}^{extd3} = \begin{cases} 1 & \text{if a driving time extension is used on arc } (i, i+1) \text{ after the last daily rest period} \\ 0 & \text{otherwise} \end{cases} \]

\[ i = 0, \ldots, r - 2 \]

\[ \mu_i^{extd} = \begin{cases} 1 & \text{if a driving time extension is decided in vertex } i \\ 0 & \text{otherwise} \end{cases} \]

\[ i = 0, \ldots, r - 1 \]

\[ \nu_i^{extd} = \begin{cases} 1 & \text{if a decision concerning a driving time extension was made before leaving vertex } i \\ 0 & \text{otherwise} \end{cases} \]

\[ i = 0, \ldots, r - 1 \]

Auxiliary variables:

\[ \lambda_i^1, \lambda_i^2, \lambda_i^3, \lambda_i^4, \lambda_i^5 \in \{0, 1\}, \quad i = 0, \ldots, r - 1 \]

\[ \lambda_i^5 \in \{0, 1\}, \quad i = 0, \ldots, r - 2 \]
2.5.4. Optional rules

There are four optional rules that provide more flexibility to the driver schedule considering a planning period of one week (see page 20). According to the EC regulation, breaks and rest periods may be split into two parts (optional rules 1 and 2). The daily driving time may be extended from 9 to 10 hours two times a week (optional rule 4). The duration of a daily rest period may be reduced to at least 9 hours at most 3 times between two weekly rest periods (optional rule 3). Each possibility has to be considered when modeling the driver activities and determining the resulting driver status when entering and leaving a vertex.

The impact of each optional rule and the associated variables and constraints will be described next in more detail. This section will end with a description and modeling of dependencies of the different optional rules.

Splitting breaks

A break may be split into two parts, the first having a duration of at least 15 minutes (first partial break), and the second (second partial break) having a duration of at least 30 minutes. After the second part of the break, a new driving time interval of at most 4.5 hours starts.

Without loss of generality we assume that if a break is split in an optimal solution, the first partial break is taken to compensate waiting time at a customer location, i.e. we only allow first partial breaks in vertices and not on arcs. If we allowed a first partial break to be taken on an arc, either it could have been postponed to the next vertex or the second part of the break would also have to be taken on the same arc without impact on the driver status. Nevertheless, in practice, a complete break scheduled between two consecutive customer locations may be split by the driver without influencing the schedule.

We introduce the binary variables $\alpha_i^{\text{break}}$ to indicate if a first partial break is taken upon arrival at a customer location $i$.

Taking into account the driver status and the driving time needed to visit the next customer stop, a break may be necessary on the way from stop $i$ to stop $i+1$. If a first partial break has already taken place, only a second partial break needs to be scheduled. The variable $\mu_{i,i+1}^{\text{break}}$ indicates that a second partial break is scheduled on arc $(i, i+1)$, i.e. the first break scheduled on this arc has a duration of 30 minutes instead of 45 minutes.

In case no break and no daily rest period are needed to traverse the arc $(i, i+1)$, a second partial break may also be taken on one of the following arcs. A second partial break may also be scheduled in vertex $i+1$ to again compensate for waiting time and to allow for a new driving time interval to start. The binary variable $\mu_i^{\text{break}}$ indicates whether a second partial break is made in vertex $i$. If no second partial break is scheduled, neither on arc $(i, i+1)$ nor in vertex $i+1$, the break may be completed on a subsequent arc or in a subsequent vertex and so on. To recall that a first partial break still may be used, we
introduce the variable $l_i^{\text{break}}$ for each customer vertex. If $l_i^{\text{break}}$ equals one, the next break will only have to take 30 minutes (second part) instead of 45 minutes.

In detail, the conditions are as follows.

Only the second partial break has to be taken on arc $(i, i + 1)$ instead of a full 45-minute break ($\mu_{(i,i+1)}^{\text{upbreak}} = 1$) if and only if a first partial break was taken before and no second partial break was made, yet (i.e. $l_i^{\text{break}} = 1$). This means that we wish to impose that

$$\mu_{(i,i+1)}^{\text{upbreak}} = 1 \iff l_i^{\text{break}} = 1 \land \alpha_{(i,i+1)}^{\text{break}} = 1$$

This is ensured by the following conditions:

\[
\begin{align*}
(\mu_{(i,i+1)}^{\text{upbreak}} = 1 & \Rightarrow l_i^{\text{break}} = 1) \\
(\mu_{(i,i+1)}^{\text{upbreak}} = 1 & \Rightarrow \alpha_{(i,i+1)}^{\text{break}} = 1) \\
(\mu_{(i,i+1)}^{\text{upbreak}} = 0 & \Rightarrow \alpha_{(i,i+1)}^{\text{break}} = 0 \lor l_i^{\text{break}} = 0)
\end{align*}
\]

Note that the last implication is the equivalent contraposition of $(l_i^{\text{break}} = 1 \land \alpha_{(i,i+1)}^{\text{break}} = 1 \Rightarrow \mu_{(i,i+1)}^{\text{upbreak}} = 1)$. For our MILP model, we obtain the following constraints:

\[
\begin{align*}
\mu_{(i,i+1)}^{\text{upbreak}} & \leq l_i^{\text{break}} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.1) \\
\mu_{(i,i+1)}^{\text{upbreak}} & \leq \alpha_{(i,i+1)}^{\text{break}} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.2) \\
\mu_{(i,i+1)}^{\text{upbreak}} & \geq l_i^{\text{break}} + \alpha_{(i,i+1)}^{\text{break}} - 1 \quad \forall i = 0, \ldots, r - 2 \quad (2.5.3)
\end{align*}
\]

A second partial break can also be taken upon the arrival at a customer location if the partial break status at the preceding vertex equals one and no break was taken on the preceding arc. Therefore, we want to state that

$$\mu_{i+1}^{\text{upbreak}} = 1 \iff l_i^{\text{break}} = 1 \land \mu_{(i,i+1)}^{\text{upbreak}} = 0 \land \alpha_{i+1}^{\text{break}} = 1$$

This is achieved by adding the following constraints, again making use of a contraposition for the formulation of the last set of constraints:

\[
\begin{align*}
\mu_{i+1}^{\text{upbreak}} & \leq l_i^{\text{break}} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.4) \\
\mu_{i+1}^{\text{upbreak}} & \leq 1 - \mu_{(i,i+1)}^{\text{upbreak}} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.5) \\
\mu_{i+1}^{\text{upbreak}} & \leq \alpha_{i}^{\text{break}} \quad \forall i = 0, \ldots, r - 1 \quad (2.5.6) \\
\mu_{i+1}^{\text{upbreak}} & \geq l_i^{\text{break}} - \mu_{(i,i+1)}^{\text{upbreak}} + \alpha_{i+1}^{\text{break}} - 1 \quad \forall i = 0, \ldots, r - 2 \quad (2.5.7)
\end{align*}
\]

Note that inequalities (2.5.4) to (2.5.7) also hold for vertex 0. The status variable $l_i^{\text{break}}$
indicating whether a first partial break still counts for the subsequent arc can now easily be determined by the following set of constraints.

\[ l_{pbreak}^{i+1} = l_{pbreak}^i - \mu_{upbreak}^{i(i+1)} - \mu_{upbreak}^{i+1} + \alpha_{pbreak}^{i+1} \quad \forall \quad i = 0, \ldots, r - 2 \] (2.5.8)

For determining the status variable for partial breaks for the first vertex, we need to know if a partial break already took place. The input parameter \( hpb \) serves as an indicator if a partial break already took place before the starting time of the schedule. It is needed when the schedule does not start at the beginning of a week (i.e. after a weekly rest period), and it is equal to one if a partial break was already taken and zero otherwise. If \( hpb \) is equal to one, a second partial break may be scheduled in the starting vertex. In that case, \( \mu_{upbreak}^0 \) is equal to one. If \( hpb \) is equal to zero, a first partial break may be taken in vertex one. Observe that \( \alpha_{pbreak}^0 \) and \( \mu_{upbreak}^0 \) cannot be both equal to one due to (2.5.6) and the vertex activity constraints (2.5.99) (see page 63).

The idea now is to define \( l_{pbreak}^i \) in such a way that a first partial break (except for a partial break taken before the planning horizon starts) always has to be exploited in the course of time. As it is possible that a first partial break has been taken before the beginning of the planning horizon, maybe it will not be beneficial to force its use. This would for example be the case if a daily rest period would be necessary or advantageous to be taken before or instead of the next break. Therefore, only an upper and a lower bound for \( l_{pbreak}^0 \) are given which induce the following logical conditions:

\[ ((hpb = 0 \land \alpha_{pbreak}^0 = 0) \lor \mu_{upbreak}^0 = 1) \Rightarrow l_{pbreak}^0 = 0 \]
\[ \alpha_{pbreak}^0 = 1 \Rightarrow l_{pbreak}^0 = 1 \]

The corresponding upper and lower bounds are given by:

\[ l_{pbreak}^0 \leq hpb + \alpha_{pbreak}^0 \] (2.5.9)
\[ l_{pbreak}^0 \leq 1 - \mu_{upbreak}^0 \] (2.5.10)
\[ l_{pbreak}^0 \geq \alpha_{pbreak}^0 \] (2.5.11)

We ensure that a first partial break is always accompanied by a second partial break to only keep track of first partial breaks that are really necessary. \( l_{pbreak}^i \) has been introduced to allow us to not only consider just the arc or vertex directly after a first partial break has been taken. If \( l_{pbreak}^i = 1 \) and no second partial break is necessary on arc \((i, i+1)\) or at vertex \(i+1\), \( l_{pbreak}^{i+1}\) again is equal to one and a second partial break is possible on arc \((i+1, i+2)\) or in vertex \(i+2\) if a break is needed, and so on.

We can distinguish between four cases for which a first partial break will be of no use, as the second part will never be scheduled.

Case 1: It is not necessary to schedule a break to fully exploit the daily driving time left and no use is made of a driving time extension. The binary variable \( l_{bn}^i \) indicates if a
break is necessary to completely use up the daily driving time left (without extension) when leaving vertex $i$. The binary variable $\mu_{(i,i+1)}^{extd1}$ indicates if a driving time extension is planned to take place on arc $(i, i+1)$ before the first daily rest period. Observe that in that case an additional break would be necessary. We obtain the following logical conditions:

\[
\begin{align*}
    l_i^{bn} = 0 & \land \mu_{(i,i+1)}^{extd1} = 0 \Rightarrow l_i^{break} = 0 \\
    l_{r-1}^{break} &= 0
\end{align*}
\]

We add the following constraints to our model.

\[
\begin{align*}
    l_i^{break} &\leq l_i^{bn} + \mu_{(i,i+1)}^{extd1} \quad \forall i = 0, \ldots, r-2 \\
    l_{r-1}^{break} &= 0
\end{align*}
\]  

(2.5.12)

(2.5.13)

Case 2: A break is replaced by a daily rest period on arc $(i, i+1)$ and this rest is the first one on this arc (i.e. $\mu_{(i,i+1)}^{earlydr1} = 1$), which means, there will be no break before the next daily rest period is taken. Hence, the second partial break again would never be scheduled.

The logical expression

\[
\mu_{(i,i+1)}^{earlydr1} = 1 \Rightarrow l_i^{break} = 0
\]

is transformed into

\[
\begin{align*}
    l_i^{break} &\leq 1 - \mu_{(i,i+1)}^{earlydr1} \quad \forall i = 0, \ldots, r-2
\end{align*}
\]  

(2.5.14)

Cases 3 and 4 refer to those situations in which a daily rest period ($\alpha_i^{rest} = 1$) or a partial daily rest period ($\alpha_i^{prest} = 1$) in vertex $i$ avoids scheduling a second partial break. Observe that even if $\alpha_i^{prest} = 0$, in case that the last break on arc $(i-1, i)$ is substituted by a partial rest (indicated by variable $\mu_i^{prest} = 1$)\(^19\), the status variable $l_{i}^{break}$ may still obtain the value 1, as a first partial break may be scheduled for a driving time extension. We transform the logical conditions

\[
\begin{align*}
    \alpha_i^{rest} = 1 &\Rightarrow l_i^{break} = 0 \land \\
    (\alpha_i^{prest} = 1 \land \mu_i^{prest} = 0) &\Rightarrow l_i^{break} = 0
\end{align*}
\]

into the following constraints:

\[
\begin{align*}
    l_i^{break} &\leq 1 - \alpha_i^{rest} \quad \forall i = 0, \ldots, r-2 \\
    l_i^{break} &\leq 1 - \alpha_i^{prest} + \mu_i^{prest} \quad \forall i = 0, \ldots, r-2
\end{align*}
\]  

(2.5.15)

(2.5.16)

\(^{19}\) The partial rest period is taken instead of the last break on the arc $(i-1, i)$ and not in vertex $i$ if $\mu_i^{prest} = 1$.  

### 2.5. Mathematical formulation

**Splitting daily rest periods**

A daily rest period may be split into two parts, one having a duration of at least 3 hours (first partial rest), and the other (second partial rest) with a duration of at least 9 hours. After the second partial rest, the daily driving time left is reset to 9 hours.

Splitting a daily rest period is advantageous if the first partial rest period compensates waiting time at a customer location and is similar to splitting breaks (see Figure 2.8, (2)).

![Splitting daily rest periods](image)

**Figure 2.8.: Splitting daily rest periods**

In addition, instead of scheduling a first partial rest period in vertex $i$, it is also possible to substitute the last break on the preceding arc by a first partial rest period ($\mu_i^{\text{prest}} = 1$). As a partial rest period also resets the 4.5 hour driving time interval, a 45 minute break can then be left out (see Figure 2.8, (3)). We will consider this in more detail at the end of this section (see constraints (2.5.24) to (2.5.30) for $\mu_i^{\text{prest}}$).

We introduce the binary variable $\alpha_i^{\text{prest}}$ to indicate if a partial rest is made upon arrival at customer location $i$. If $\alpha_i^{\text{prest}} = 1$, then it may be necessary to schedule the second partial rest on the way from location $i$ to location $i + 1$. The driver status as well as the driving time needed to travel from $i$ to $i + 1$ determine whether a daily rest period is required on the arc $(i, i + 1)$. In case no daily rest period is needed, the first partial rest still "counts"
for the subsequent arc. To keep track if a first partial rest still may be used, we introduce the binary variable $l_{i}^{\text{prest}}$. If $l_{i}^{\text{prest}} = 1$ then the next daily rest period will only have a duration of at least 9 hours instead of 11 hours.

In contrast to the case of splitting breaks, it is not necessary to define a variable to indicate if a second partial rest period is scheduled. As sometimes it may be beneficial to take a daily rest period that lasts longer than the minimum duration of 9 or 11 hours, respectively, variables are introduced to identify the actual duration instead (see pages 58-59, constraints (2.5.79) to (2.5.84)).

Also different from partial breaks, it may be advantageous to take a first partial rest for which the second part will be scheduled after the planning horizon as the decision about taking a first partial rest will influence the time left until the next daily rest period is necessary. If, for example, a first partial rest is taken at the last customer location, the remaining duration of the second partial rest period is at least 9 hours instead of 11 hours. In that case, more time will be left to fulfill the service at the customer location.20

Let us now consider the status variable $l_{i}^{\text{prest}}$, which indicates whether only the second part of a daily rest period is still needed to reset the time left until the next daily rest period. If a daily rest period is taken in vertex $i$, the status variable $l_{i}^{\text{prest}}$ is set to zero as a potential first partial rest would have been used to reduce the duration of this daily rest period. Hence,

$$\alpha_{i}^{\text{rest}} = 1 \Rightarrow l_{i}^{\text{prest}} = 0$$

which is expressed by

$$l_{i}^{\text{prest}} \leq 1 - \alpha_{i}^{\text{rest}} \quad \forall \ i = 0, \ldots, r - 1$$

(2.5.17)

Clearly, $l_{i}^{\text{prest}}$ should always be equal to 1 if a first partial rest is taken in vertex $i$. That means

$$\alpha_{i}^{\text{prest}} = 1 \Rightarrow l_{i}^{\text{prest}} = 1$$

This is induced by

$$l_{i}^{\text{prest}} \geq \alpha_{i}^{\text{prest}} \quad \forall \ i = 0, \ldots, r - 1$$

(2.5.18)

If a daily rest period is taken on arc $(i, i + 1)$, the status variable $l_{i+1}^{\text{prest}}$ is only influenced

---

20 Assume that in Figure 2.8 the location $i + 1$ is the last customer location and no reduced daily rest period is possible anymore (i.e. in the current week, 3 reduced daily rest periods have already taken place). The driver would reach the time window, but adding 2 hours for serving the customer would lead to a working time of 13:30 h since the last daily rest period. This violates rule 6 as it is not possible to append a daily rest period of 11 h and finish it within the 24 h time interval. A daily rest period would be necessary before loading and/or unloading the vehicle which would have to be postponed to the following day.
by a potential partial rest in vertex \( i + 1 \). We would like to state

\[
\alpha_{i,(i+1)}^{\text{rest}} = 1 \Rightarrow l_{i+1}^{\text{prest}} = \alpha_{i+1}^{\text{prest}}
\]

This is imposed by inequalities (2.5.18) and the following set of constraints:

\[
l_{i+1}^{\text{prest}} \leq \alpha_{i+1}^{\text{prest}} + 1 - \alpha_{i,(i+1)}^{\text{rest}} \quad \forall \ i = 0, \ldots, r - 2
\]  

(2.5.19)

An upper bound on the value of the status variable \( l_{i+1}^{\text{prest}} \) is \( l_{i+1}^{\text{prest}} = \alpha_{i+1}^{\text{prest}} \). If both variables \( l_{i+1}^{\text{prest}} \) and \( \alpha_{i+1}^{\text{prest}} \) are zero then \( l_{i+1}^{\text{prest}} \) must also be equal to zero:

\[
l_{i}^{\text{prest}} = 0 \land \alpha_{i+1}^{\text{prest}} = 0 \Rightarrow l_{i+1}^{\text{break}} = 0
\]

This is induced by

\[
l_{i+1}^{\text{prest}} \leq l_{i}^{\text{prest}} + \alpha_{i+1}^{\text{prest}} \quad \forall \ i = 0, \ldots, r - 2
\]  

(2.5.20)

If neither a daily rest period is taken on arc \((i, i+1)\) nor in vertex \( i + 1 \), then \( l_{i+1}^{\text{prest}} \) depends on both variables, \( l_{i}^{\text{prest}} \) and \( \alpha_{i+1}^{\text{prest}} \). In this case, we have

\[
(\alpha_{i,(i+1)}^{\text{rest}} = 0 \land \alpha_{i+1}^{\text{rest}} = 0) \Rightarrow (l_{i+1}^{\text{prest}} = 1 \iff l_{i}^{\text{prest}} = 1 \lor \alpha_{i+1}^{\text{prest}} = 1)
\]

This is ensured by constraints (2.5.20) and

\[
l_{i+1}^{\text{prest}} \geq l_{i}^{\text{prest}} + \alpha_{i+1}^{\text{prest}} - \alpha_{i,(i+1)}^{\text{rest}} - \alpha_{i+1}^{\text{rest}} \quad \forall \ i = 0, \ldots, r - 2
\]  

(2.5.21)

Note that inequalities (2.5.21) enforce \( l_{i}^{\text{prest}} + \alpha_{i+1}^{\text{prest}} \leq 1 \) in the case that no daily rest period on the arc \((i, i+1)\) or in the vertex \( i \) is taken. This means that a first partial rest period has to be exploited by taking a second partial daily rest period before a new first partial daily rest period can be taken.

In the first vertex, \( l_{0}^{\text{prest}} \) only depends on \( \alpha_{0}^{\text{prest}} \) and if a partial rest was taken before the start of the planning horizon \((hpr = 1)\) if no rest is taken in \( 0 \). The case that a rest is taken is already covered by constraints (2.5.17).

\[
\alpha_{0}^{\text{rest}} = 0 \Rightarrow l_{0}^{\text{prest}} = \alpha_{0}^{\text{prest}} + hpr
\]

As \( \alpha_{0}^{\text{prest}} + hpr \) is an upper bound on \( l_{0}^{\text{prest}} \), we obtain the following constraints:

\[
l_{0}^{\text{prest}} \geq \alpha_{0}^{\text{prest}} + hpr - \alpha_{0}^{\text{rest}}
\]  

(2.5.22)

\[
l_{0}^{\text{prest}} \leq \alpha_{0}^{\text{prest}} + hpr
\]  

(2.5.23)
As mentioned at the beginning of this section instead of taking a break as the last resting activity on arc \((i, i+1)\), it may also be possible to substitute this break by a first partial rest period. The first partial rest also resets the driving time left until the next break or rest period, so it may save time to just schedule a first partial rest period on the arc \((i, i+1)\) as the last resting activity instead of a 45-minute break and additionally a first partial daily rest period in vertex \(i+1\) (see Figure 2.8 (3)). We introduce the binary variable \(\mu^\text{prest}_i\) to indicate that a substitution takes place and the status variables when entering vertex \(i\) and the arrival time at customer \(i\) have to be modified accordingly. If such a substitution is planned, a first partial rest period has to be scheduled:

\[
\mu^\text{prest}_i = 1 \Rightarrow \alpha^\text{prest}_i = 1
\]

This logical condition is represented by

\[
\alpha^\text{prest}_i \geq \mu^\text{prest}_i \quad \forall \ i = 1, \ldots, r - 1 \quad (2.5.24)
\]

For vertex 0, since there is no preceding arc where we can substitute a break, we have

\[
\mu^\text{prest}_0 = 0 \quad (2.5.25)
\]

For other vertices \(i\), there has to be a break on arc \((i-1, i)\) that may be substituted and this break has to be the last resting activity on this arc. When entering vertex \(i\), variable \(\epsilon^\text{bt}_i\) indicates if the last rest activity taken on the arc \((i-1, i)\) was a break. We obtain the following upper bounds on \(\mu^\text{prest}_i\):

\[
\mu^\text{prest}_i \leq \alpha^\text{break}_{(i-1, i)} \quad \forall \ i = 1, \ldots, r - 1 \quad (2.5.26)
\]

\[
\mu^\text{prest}_i \leq \epsilon^\text{bt}_i \quad \forall \ i = 1, \ldots, r - 1 \quad (2.5.27)
\]

If the daily driving time should be extended and the corresponding resting activity is planned to be a first partial rest in a vertex, a preceding break on arc \((i, i+1)\) will not be substituted, as both rest activities (45-minute break and first partial rest) are needed to be able to extend the daily driving time to 10 hours. Therefore, we obtain the upper bounds:

\[
\mu^\text{prest}_i \leq 1 - \mu^\text{extd}_i \quad \forall \ i = 1, \ldots, r - 1 \quad (2.5.28)
\]

Whenever possible, the substitution should take place, as it will never worsen the objective function value and it may save up to 45 minutes of time. Whenever possible means, if a break is made on arc \((i-1, i)\) \((\alpha^\text{break}_{(i-1, i)} = 1)\), the last resting activity on this arc was a break, and a first partial rest is associated with vertex \(i\) \((\alpha^\text{prest}_i = 1)\), then the break will be substituted if no break for a driving time extension is needed in vertex \(i\). This is expressed by

\[
(\alpha^\text{break}_{(i-1, i)} = 1 \land \epsilon^\text{bt}_i = 1 \land \mu^\text{extd}_i = 0) \Rightarrow \mu^\text{prest}_i = \alpha^\text{prest}_i
\]
2.5. Mathematical formulation

and guaranteed by inequalities (2.5.24) and the following constraints

\[ \mu_i^{\text{prest}} \geq \alpha_i^{\text{prest}} - \mu_i^{\text{extd}} - (1 - \alpha_{i-1,i}) - (1 - \epsilon_i^{\text{bt}}) \quad \forall \ i = 1, \ldots, r - 1 \]  

(2.5.29)

If there is only one break on arc \((i-1, i)\), a potential first partial break still counting when leaving vertex \(i-1\) may be consumed by the substitution and would therefore not be necessary to be scheduled. To avoid this, we add the following upper bound on \(l_{i-1}^{\text{break}}\) with \(A_{(i-1,i)}^{\text{break}}\) being the number of breaks scheduled on arc \((i-1, i)\):

\[ l_{i-1}^{\text{break}} \leq A_{(i-1,i)}^{\text{break}} - \mu_i^{\text{prest}} + (1 - \alpha_{i-1,i}^{\text{break}}) \quad \forall \ i = 1, \ldots, r - 1 \]  

(2.5.30)

**Reducing daily rest periods**

A daily rest period may be reduced from 11 to 9 hours at most three times a week. If it is reduced, the time between the end of a daily rest period and the beginning of the subsequent reduced daily rest period is automatically extended from 13 to 15 hours because of standard rule 6 (page 19), which states that there has to be a daily rest period in each time interval with a duration of 24 hours.

The difficulty here lies in the fact that after a daily rest period, depending on whether the next daily rest period is a reduced one or not, the time interval between the two rest periods has a duration of 13 or 15 hours, respectively.

To overcome this, for the first daily rest period on an arc, we introduce the variable \(\mu_i^{\text{dredrest}}\) associated with the decision about the next rest after leaving vertex \(i\) being a reduced one. Additionally, the status variable \(l_{i}^{\text{dredrest}}\) monitors if a decision about a reduced daily rest period was made before leaving vertex \(i\) to keep track of a decision made about a reduced daily rest period at a vertex prior to \(i\) if no rest period was scheduled since then.

If a second or third daily rest period is taken on an arc between two customers, only the driving time has to be considered. As there are always at least 13 hours between two daily rest periods but only at most 10 hours of driving allowed, no special care has to be taken about a reduced daily rest period. However, its duration needs to be modified.

The variable \(\mu_{(i,i+1)}^{\text{redrest}}\) gives the number of reduced rest periods scheduled for the arc \((i, i+1)\), while the variable \(\mu_i^{\text{redrest}}\) indicates if a reduced rest period is scheduled in vertex \(i\). These variables are used to modify the duration of daily rest periods such that instead of 11 hours at least 9 hours are needed.\(^{21}\) This means that the reduction has to be scheduled together with the daily rest period itself, namely

\[ (\mu_i^{\text{redrest}} = 1 \Rightarrow \alpha_i^{\text{rest}} = 1) \quad \land \quad (\mu_{(i,i+1)}^{\text{redrest}} = k \Rightarrow A_{(i,i+1)}^{\text{rest}} \geq k), \quad k \in \{0, 1, 2, 3\} \]

\(^{21}\) Restrictions on the duration of daily rest periods are described in more detail in Section 2.5.9 on page 58.
The above conditions are expressed by constraints (2.5.31) and (2.5.32).

\[
\alpha_i^{rest} \geq \mu_i^{redrest} \quad \forall i = 0, \ldots, r - 1 \quad (2.5.31)
\]

\[
A_{(i,i+1)}^{rest} \geq \mu_{(i,i+1)}^{redrest} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.32)
\]

If when leaving vertex \(i\) the decision is made that the next daily rest period will be a reduced one \(\mu_i^{redrest} = 1\), the corresponding status variable \(l_i^{redrest}\) is set to be equal to one. This means that

\[
\mu_i^{redrest} = 1 \Rightarrow l_i^{redrest} = 1
\]

and this is represented by

\[
l_i^{redrest} \geq \mu_i^{redrest} \quad \forall i = 0, \ldots, r - 1 \quad (2.5.33)
\]

If a daily rest period is taken either on arc \((i - 1, i)\) or in vertex \(i\), then (2.5.33) should hold as an equality, since in this case the status variable \(l_i^{redrest}\) only depends on the decision about a reduced rest period when leaving vertex \(i\). The execution of former decisions about short rests lies in the past. This can be stated by

\[
(\alpha_{(i-1,i)}^{rest} = 1 \Rightarrow l_i^{redrest} = \mu_i^{redrest}) \land (\alpha_i^{rest} = 1 \Rightarrow l_i^{redrest} = \mu_i^{redrest})
\]

This is induced by constraints (2.5.33), (2.5.34) and (2.5.35)

\[
l_i^{redrest} \leq \mu_i^{redrest} + (1 - \alpha_{(i-1,i)}^{rest}) \quad \forall i = 1, \ldots, r - 1 \quad (2.5.34)
\]

\[
l_i^{redrest} \leq \mu_i^{redrest} + (1 - \alpha_i^{rest}) \quad \forall i = 0, \ldots, r - 1 \quad (2.5.35)
\]

In case neither a daily rest period was made on the arc \((i, i + 1)\) nor in vertex \(i + 1\), the status variable \(l_{i+1}^{redrest}\) additionally depends on the status variable \(l_i^{redrest}\) of the preceding vertex \(i\). Hence,

\[
(\alpha_i^{rest} = 0 \land \alpha_{i+1}^{rest} = 0) \Rightarrow (l_{i+1}^{redrest} = l_i^{redrest} + \mu_{i+1}^{redrest})
\]

As \(l_i^{redrest} + \mu_{i+1}^{redrest}\) is an upper bound on \(l_{i+1}^{redrest}\), we obtain the following constraints:

\[
l_{i+1}^{redrest} \leq l_i^{redrest} + \mu_{i+1}^{redrest} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.36)
\]

\[
l_{i+1}^{redrest} \geq l_i^{redrest} + \mu_{i+1}^{redrest} - \alpha_i^{rest} - \alpha_{(i,i+1)}^{rest} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.37)
\]

The status variable \(l_0^{redrest}\) of the starting vertex only depends on the decision about a reduced daily rest period:

\[
l_0^{redrest} = \mu_0^{redrest} \quad (2.5.38)
\]
If several daily rest periods are scheduled on an arc \((i, i + 1)\), the maximum time saving with a reduced daily rest period can be made by planning the first daily rest period to be a reduced one. Therefore, in case at least one reduced daily rest period is planned for \((i, i + 1)\), we set \(l_i^{\text{dredrest}} = 1\), except for the case a second partial rest period still has to be scheduled. Similarly, if a reduced daily rest period is taken in vertex \(i + 1\) and there were no daily rest periods on the previous arc \((i, i + 1)\), the status variable \(l_i^{\text{dredrest}}\) is set to be equal to 1 if \(l_i^{\text{prest}} = 0\). The corresponding logical conditions are:

\[
\begin{align*}
\left( \mu_i^{\text{redrest}} \geq 1 \land p_i^{\text{prest}} = 0 \right) & \implies l_i^{\text{dredrest}} = 1 \land \\
\left( \mu_{i+1}^{\text{redrest}} = 1 \land \alpha_{i,i+1}^{\text{rest}} = 0 \right) & \implies l_i^{\text{dredrest}} = 1
\end{align*}
\]

The total number of reduced daily rest periods in one week is at most 3 and therefore, this is also an upper bound on the number of reduced daily rest periods on an arc. It follows that the above conditions are induced by

\[
3 l_i^{\text{dredrest}} \geq \mu_i^{\text{redrest}} - p_i^{\text{prest}} \quad \forall \ i = 0, \ldots, r - 2 \tag{2.5.39}
\]

\[
l_i^{\text{dredrest}} \geq \mu_i^{\text{redrest}} - \alpha_{i,i+1}^{\text{rest}} \quad \forall \ i = 0, \ldots, r - 2 \tag{2.5.40}
\]

Additionally, we interlink the decision about a reduced daily rest period and the end of the last daily rest period to ensure that the status variables \(E_t^i, L_t^i, L_{dd}^i\) and \(L_{dt}^i\) for vertex \(i\) reflect the exact driver status. The decision that the next daily rest period should be a reduced one \(\mu_{i+1}^{\text{dredrest}} = 1\) can only be made if there was a daily rest period on the previous arc or vertex. This condition guarantees that decisions about reduced daily rest periods are made as early as possible such that status variables will reflect the actual driver status when entering or leaving a vertex. We add the conditions

\[
(\alpha_{i,i+1}^{\text{rest}} = 0 \land \alpha_{i+1}^{\text{rest}} = 0) \implies \mu_{i+1}^{\text{dredrest}} = 0
\]

and obtain the following set of inequalities:

\[
\alpha_{i,i+1}^{\text{rest}} \leq \alpha_i^{\text{rest}} + \alpha_{i+1}^{\text{rest}} \quad \forall \ i = 0, \ldots, r - 2. \tag{2.5.41}
\]

Reduced rests can now be scheduled according to the status variables \(l_i^{\text{dredrest}}\). That means, if at least one daily rest period is scheduled on an arc \((i, i + 1)\) and the status variable \(l_i^{\text{dredrest}}\) is equal to 1, then at least one of the daily rest periods has to be a reduced one.

\[
\alpha_{i,i+1}^{\text{rest}} = 1 \implies \mu_{i,i+1}^{\text{dredrest}} \geq l_i^{\text{dredrest}}
\]

\[\text{Note that reduced daily rest periods may be scheduled on arc } (i, i + 1) \text{ if } l_i^{\text{dredrest}} = 0. \text{ In that case, additional time prior to this reduced daily rest period is not needed, just the 2 hours time saving for the reduction of the duration is considered.}\]
This is ensured by
\[
\mu_{(i,i+1)}^{\text{redrest}} \geq \mu_i^{\text{dredrest}} - (1 - \alpha_{(i,i+1)}^{\text{rest}}) \quad \forall \ i = 0, \ldots, r - 2
\] (2.5.42)

If no rest is made on the arc \((i, i+1)\), a similar condition holds for the following vertex \(i + 1\):
\[
\alpha_{(i,i+1)}^{\text{rest}} = 0 \Rightarrow (\alpha_{i+1}^{\text{rest}} = 1 \Rightarrow \mu_{i+1}^{\text{redrest}} \geq t_i^{\text{dredrest}})
\]

This is guaranteed by the following constraints.
\[
\mu_{i+1}^{\text{redrest}} \geq \mu_i^{\text{dredrest}} - \alpha_{(i,i+1)}^{\text{rest}} - (1 - \alpha_{i+1}^{\text{rest}}) \quad \forall \ i = 0, \ldots, r - 2
\] (2.5.43)

To ensure that the maximum number of reduced daily rest periods during one week is not exceeded, we add the constraint
\[
\sum_{i=0}^{r-1} \mu_i^{\text{redrest}} + \sum_{i=0}^{r-2} \mu_{(i,i+1)}^{\text{redrest}} + \overline{\text{noRed}} \leq 3
\] (2.5.44)

where \(\overline{\text{noRed}}\) is the number of reduced daily rest periods already taken in the current week in the time before the start of the schedule.

We conclude this section by describing constraints that map dependencies between partial and reduced daily rest periods. First, we have to ensure that enough rest periods are scheduled, one for each decision about a reduced daily rest period and one for each second partial rest period. For the case that rest periods are taken on arc \((i, i+1)\), we obtain the condition
\[
\alpha_{(i,i+1)}^{\text{rest}} = 1 \Rightarrow A_{(i,i+1)}^{\text{rest}} \geq \mu_{(i,i+1)}^{\text{redrest}} + t_i^{\text{p pret}}
\]

We transform this into
\[
\mu_{(i,i+1)}^{\text{redrest}} + t_i^{\text{p pret}} \leq A_{(i,i+1)}^{\text{rest}} + (1 - \alpha_{(i,i+1)}^{\text{rest}}) \quad \forall \ i = 0, \ldots, r - 2
\] (2.5.45)

In addition, only one of the variables, \(t_i^{\text{dredrest}}\) or \(t_i^{\text{p pret}}\), may take on the value 1. Both cannot be equal to 1 as otherwise, the next daily rest period would be a second partial rest period and a reduced daily rest period as well, which does not make any sense.

\[
(t_i^{\text{dredrest}} = 1 \Rightarrow t_i^{\text{p pret}} = 0) \quad \land \quad (t_i^{\text{p pret}} = 1 \Rightarrow t_i^{\text{dredrest}} = 0)
\]

is imposed by
\[
\mu_i^{\text{dredrest}} + t_i^{\text{p pret}} \leq 1 \quad \forall \ i = 0, \ldots, r - 1
\] (2.5.46)
For the first vertex it has to be ensured that no reduced rest period is scheduled if a second partial rest is still outstanding. Hence,

\[ h_{pr} = 1 \Rightarrow \mu_0^{redrest} = 0 \]

This is guaranteed by the following constraint.

\[ \mu_0^{redrest} \leq 1 - h_{pr} \]  \hspace{1cm} (2.5.47)

**Extending daily driving times**

The daily driving time, i.e. the cumulated driving time between two consecutive daily rest periods, may be extended from 9 to 10 hours twice a week. If a daily driving time is extended, an additional break (or a first partial daily rest period) will be necessary, as there has to be a break after at most 4.5 hours of driving.

Driving time extensions have an impact on the values of the status variables \( E^t_{i+1}, L^t_{i+1}, E^{ddt}_{i+1}, L^{ddt}_{i+1} \) and \( E^{dt}_{i+1}, L^{dt}_{i+1} \) depending on when the additional break (or first partial daily rest period) is taken. We distinguish between the following four cases and introduce decision variables accordingly. The additional break can be taken:

- **case 1:** in a vertex \( i \) (\( \mu_{extd}^i = 1 \)),
- **case 2:** on an arc \((i, i+1)\), before the first daily rest period is taken or if no daily rest period is taken on this arc (\( \mu_{extd1}^{i,i+1} = 1 \)),
- **case 3:** on an arc \((i, i+1)\), between two consecutive daily rest periods (\( \mu_{extd2}^{i,i+1} = 1 \)) or
- **case 4:** on an arc \((i, i+1)\), after the last daily rest period (\( \mu_{extd3}^{i,i+1} = 1 \)).

The main difference between scheduling a break or a first partial rest period for a driving time extension on an arc from scheduling it in a vertex is that they are always scheduled as late as possible (i.e. after 4.5 hours of driving, directly initiating the extension). In general, breaks can be scheduled in vertices to reduce or avoid waiting time, and therefore they may be taken before the limit for the driving time without break or rest period of 4.5 hours is reached.

Furthermore, on an arc a break or a first daily rest period for a driving time extension can be taken before the first daily rest period is completed (if there is a daily rest period on this arc) (case 1), between two consecutive daily rest periods (case 2) or after the last daily rest period (case 3).

In case 1, the time left until the next rest period when leaving vertex \( i \), \( L^t_i \), has to be taken into consideration, as this may limit the extended daily driving time to a value that is less than 10 hours. If a driving time extension occurs between two consecutive daily rest periods (\( \mu_{extd2} = 1 \)) (case 2), then this will lead to the maximum daily driving time of 10 hours as the daily driving time is not limited by the maximum time interval between two
consecutive daily rest periods. For the third case, we have to ensure that two breaks after the last daily rest period are scheduled such that the last of the two breaks coincides with the decision about the driving time extension.

Hence, if a special type of driving time extension can be scheduled on an arc, it depends on the number of daily rest periods on this arc. \( \mu_{i,i+1}^{extd1} \) does not depend on whether one or more daily rest periods are made on arc \((i, i+1)\), whereas \( \mu_{i,i+1}^{extd3} = 1 \) requires at least one daily rest period. \( \mu_{i,i+1}^{extd2} \) larger than 0 requires at least two daily rest periods on arc \((i, i+1)\), depending on the number of driving time extensions taken between the first and the last daily rest period.

We wish to state that

\[
\mu_{i,i+1}^{extd2} = k \Rightarrow A_{i,i+1}^{rest} \geq k + 1 \quad \text{with } k \in \{1, 2\} \quad \text{and}
\]
\[
\mu_{i,i+1}^{extd3} = 1 \Rightarrow \alpha_{i,i+1}^{rest} = 1
\]

This is achieved by the following constraints:

\[
A_{i,i+1}^{rest} \geq \mu_{i,i+1}^{extd2} + \alpha_{i,i+1}^{rest} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.48)
\]
\[
\alpha_{i,i+1}^{rest} \geq \mu_{i,i+1}^{extd3} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.49)
\]

In addition to the decision variables for extending driving, we again need status variables which in this case indicate if a driving time extension still holds when leaving vertex \(i\) (\(l_{i}^{extd}\)). This status variable ensures that a driving time extension cannot be made twice without a rest period in between.

We start with the corresponding constraint for vertex 0. The input data tells us if an extended daily driving time has already started: if the daily driving time \(ddt\) is greater than 540 and the number of extended driving times already taken that week is less than 2, the input parameter \(dte\) is set to be equal to 1, otherwise \(dte = 0\). We require that an already started driving time remains active if no daily rest period is taken in vertex 0:

\[
\alpha_{0}^{rest} = 0 \Rightarrow l_{0}^{extd} = dte
\]

This is guaranteed by

\[
l_{0}^{extd} \geq dte - \alpha_{0}^{rest} \quad \text{and}
\]
\[
l_{0}^{extd} \leq dte \quad (2.5.50)
\]

No driving time extension needs to be scheduled in vertex 0 as such an extension can also be postponed to the arc \((0, 1)\). Hence,

\[
\mu_{0}^{extd} = 0. \quad (2.5.52)
\]
Before a daily rest period is taken on arc \((i, i + 1)\), a driving time extension can only take place \((\mu_{xtd}^{1(i,i+1)} = 1)\) if no driving time extension is still active when leaving customer \(i\). This is expressed by

\[ t_{i}^{xtd} = 1 \implies \mu_{xtd}^{1(i,i+1)} = 0 \]

We add the constraints

\[ \mu_{xtd}^{1(i,i+1)} \leq 1 - t_{i}^{xtd} \quad \forall \ i = 0, \ldots, r - 2 \] (2.5.53)

to our model.

We wish to avoid that an early daily rest period is scheduled \((\mu_{earlydr}^{1(i,i+1)} = 1)\) simultaneously with a driving time extension before the first daily rest period (if one is taken) on arc \((i, i + 1)\) is taken, since in reality this combination is impossible.\(^{23}\) Therefore, the following condition must hold.

\[ \mu_{earlydr}^{1(i,i+1)} = 1 \implies \mu_{xtd}^{1(i,i+1)} = 0 \]

This is imposed by constraints (2.5.54).

\[ \mu_{xtd}^{1(i,i+1)} \leq 1 - \mu_{earlydr}^{1(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \] (2.5.54)

Usually, it does not improve the solution value to schedule a combination of extended driving times and early daily rest periods on one arc. However, there are two exceptions: the combination of an early daily rest period as first daily rest period on arc \((i, i + 1)\) and an extended driving time of type two \((\mu_{xtd}^{2(i,i+1)} \geq 1)\) or of type three \((\mu_{xtd}^{3(i,i+1)} = 1)\). The other cases are avoided if the following conditions are imposed. The variable \(\mu_{earlydr}^{2(i,i+1)} = 1\) indicates that an early daily rest period is taken and this daily rest period is not the first one on this arc.

\[ \mu_{earlydr}^{2(i,i+1)} = 1 \implies (\mu_{xtd}^{1(i,i+1)} = 0 \land \mu_{xtd}^{2(i,i+1)} = 0 \land \mu_{xtd}^{3(i,i+1)} = 0) \]

We add the following constraints:

\[ \mu_{xtd}^{1(i,i+1)} \leq 1 - \mu_{earlydr}^{2(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \] (2.5.55)

\[ \mu_{xtd}^{2(i,i+1)} \leq 2 - 2 \mu_{earlydr}^{2(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \] (2.5.56)

\[ \mu_{xtd}^{3(i,i+1)} \leq 1 - \mu_{earlydr}^{2(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \] (2.5.57)

\(^{23}\) For more details on early daily rest periods see page 61.
In constraints (2.5.56), the upper bound 2 is the maximum number of driving time extensions allowed during one week.

A driving time extension $\mu_{i,i+1}^{\text{extd}} = 1$ may not be possible if most of the available time between two daily rest periods was already spent for other activities different from driving. In that case, $\mu_{i,i+1}^{\text{extd}}$ is set to be zero. The variable giving us information if a driving time extension is possible, considering the maximum time left until the start of the next daily rest period, is variable $\lambda_i^1$. If $\lambda_i^1$ is equal to 1, then no extension is possible, otherwise an extension is permitted:

$$\lambda_i^1 = 1 \Rightarrow \mu_{i,i+1}^{\text{extd}} = 0$$

This is enforced by

$$\mu_{i,i+1}^{\text{extd}} \leq 1 - \lambda_i^1 \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.58)$$

For $\mu_{i,i+1}^{\text{extd}}$, we wish to ensure that the driving time extension is coupled with a second break since the last daily rest period. This means that at least nine hours of daily driving time have already been consumed when entering vertex $i + 1$ since the MILP model schedules breaks and rest periods on arcs as late as possible. Therefore, at most one hour may be left for driving until the next rest period. Similarly, this is also true for $\mu_{i,i+1}^{\text{extd}}$ if no daily rest period is taken on arc $(i, i+1)$. Hence, the following conditions must hold:

$$\mu_{i,i+1}^{\text{extd}} = 1 \Rightarrow E_{i+1}^{\text{ddt}} \leq 60$$

$$(\mu_{i,i+1}^{\text{extd}} = 1 \land \alpha_{i,i+1}^{\text{rest}} = 0) \Rightarrow E_{i+1}^{\text{ddt}} \leq 60$$

With 540 (i.e. 9 h) being an upper bound on $E_{i+1}^{\text{ddt}}$, we can express this condition using a "big-M approach" with $M = 480$ ($= 540 - 60$).

$$E_{i+1}^{\text{ddt}} \leq 60 + 480 \left(1 - \mu_{i,i+1}^{\text{extd}}\right) \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.59)$$

$$E_{i+1}^{\text{ddt}} \leq 60 + 480 \left(1 - \mu_{i,i+1}^{\text{extd}}\right) + 480 \alpha_{i,i+1}^{\text{rest}} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.60)$$

If there was no rest period since leaving the last vertex (i.e. $\alpha_{i,i+1}^{\text{rest}} = 0$) and a driving time extension was active when leaving that vertex ($l_{i+1}^{\text{extd}} = 1$), then this driving time extension is still active when entering $i + 1$. As a result, another driving time extension is not allowed to start in vertex $i + 1$. A new driving time extension may not start in $i + 1$ if a driving time extension has already started on arc $(i, i+1)$ and no rest period was taken since then:

$$(l_{i}^{\text{extd}} = 1 \lor \mu_{i,i+1}^{\text{extd}} = 1) \land \alpha_{i,i+1}^{\text{rest}} = 0 \Rightarrow \mu_{i+1}^{\text{extd}} = 0$$

$^{24}$ The auxiliary variable $\lambda^1$ is described in more detail on page 76.
This is imposed by

\[ \mu_{i+1}^{extd} \leq 1 - l_{i+1}^{extd} + \alpha_{i(i+1)}^{rest} \]
\[ \forall i = 0, \ldots, r - 2 \quad (2.5.61) \]

\[ \mu_{i+1}^{extd} \leq 1 - \mu_{(i,i+1)}^{extd1} + \alpha_{i(i+1)}^{rest} \]
\[ \forall i = 0, \ldots, r - 2 \quad (2.5.62) \]

For the case that an extended driving time has started on arc \((i, i + 1)\) after the last daily rest period, i.e. \(\mu_{(i,i+1)}^{extd3} = 1\), the above conditions are completed by adding the following one:

\[ \mu_{(i,i+1)}^{extd3} = 1 \Rightarrow \mu_{i+1}^{extd} = 0 \]

We obtain the additional constraints

\[ \mu_{i+1}^{extd} \leq 1 - \mu_{(i,i+1)}^{extd3} \]
\[ \forall i = 0, \ldots, r - 2 \quad (2.5.63) \]

If the daily driving time left, \(E_{i}^{ddt}\), is higher than 270 minutes (that is 4.5 h), a break in vertex \(i\) should not be considered to extend the daily driving time:

\[ E_{i}^{ddt} > 270 \Rightarrow \mu_{i}^{extd} = 0 \]

This is ensured by the following constraints:

\[ 270 (1 - \mu_{i}^{extd}) \geq E_{i}^{ddt} - 270 \quad \forall i = 0, \ldots, r - 1 \quad (2.5.64) \]

Now, the determination of the status variables \(l_{i+1}^{extd}\) when leaving vertex \(i + 1\) is described. If a daily rest period is taken in vertex \(i\), then a new driving time interval starts. An extended driving time still active when entering vertex \(i\) is finished with the start of the daily rest period. Therefore, the status variable \(l_{i+1}^{extd}\) is set to be zero. If a daily rest period is taken on arc \((i, i + 1)\), \(l_{i+1}^{extd}\) depends on \(\mu_{i(i+1)}^{extd3}\) and \(\mu_{i+1}^{extd}\). It depends on \(l_{i}^{extd}\) and \(\mu_{(i,i+1)}^{extd1}\) and \(\mu_{i+1}^{extd}\) if no daily rest period is taken on arc \((i, i + 1)\). In particular, the following conditions have to hold:

\[ \alpha_{i}^{rest} = 1 \Rightarrow l_{i}^{extd} = 0 \quad \wedge \]
\[ (\alpha_{(i,i+1)}^{rest} = 1 \wedge \alpha_{i+1}^{rest} = 0) \Rightarrow (l_{i+1}^{extd} = \mu_{(i,i+1)}^{extd3} + \mu_{i+1}^{extd}) \quad \wedge \]
\[ (\alpha_{(i,i+1)}^{rest} = 0 \wedge \alpha_{i+1}^{rest} = 0) \Rightarrow (l_{i+1}^{extd} = l_{i}^{extd} + \mu_{(i,i+1)}^{extd1} + \mu_{i+1}^{extd}) \]

These conditions are represented by:

\[ l_{i}^{extd} \leq 1 - \alpha_{i}^{rest} \]
\[ \forall i = 0, \ldots, r - 1 \quad (2.5.65) \]

\[ l_{i+1}^{extd} \geq \mu_{i+1}^{extd} + \mu_{(i,i+1)}^{extd3} - \alpha_{i+1}^{rest} \]
\[ \forall i = 0, \ldots, r - 2 \quad (2.5.66) \]

\[ l_{i+1}^{extd} \leq \mu_{i+1}^{extd} + \mu_{(i,i+1)}^{extd3} + 1 - \alpha_{(i,i+1)}^{rest} \]
\[ \forall i = 0, \ldots, r - 2 \quad (2.5.67) \]
2. Scheduling of driving times, breaks and rest periods

\[
l^\text{extd}_{i+1} \geq l^\text{extd}_i + \mu^\text{extd}_{(i,i+1)} + \mu^\text{extd}_{i+1} - \alpha^\text{rest}_{(i,i+1)} - \alpha^\text{rest}_{i+1} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.68)
\]

\[
l^\text{extd}_{i+1} \leq l^\text{extd}_i + \mu^\text{extd}_{(i,i+1)} + \mu^\text{extd}_{i+1} + \alpha^\text{rest}_{(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.69)
\]

Inequalities (2.5.66) and (2.5.67) are lower and upper bounds on \(l^\text{extd}_{i+1}\), respectively. Therefore, \((1 - \alpha^\text{rest}_{(i,i+1)})\) needs not to be subtracted in constraints (2.5.66) and \(\alpha^\text{rest}_{i+1}\) needs not to be added in constraints (2.5.67). Similarly, \(\alpha^\text{rest}_{i+1}\) needs not to be added in constraints (2.5.69).

The number of extended driving times during the week is bounded from above by the maximum between 0 and 2 minus the number of extended driving times that were already used since the start of the week.25

\[
\sum_{i=0}^{r-2} \mu^\text{extd}_{(i,i+1)} + \mu^\text{extd}_{(i+1,i)} + \mu^\text{extd}_{i+1} \leq \max\{2 - \text{noExt} - d\text{te} - 0\} \quad (2.5.70)
\]

As mentioned at the beginning, every daily driving time extension is coupled with a break, meaning that a break has to be scheduled for each driving time extension. Hence, the following constraints have to be added:

\[
\alpha^\text{break}_i + \alpha^\text{prest}_i - \mu^\text{prest}_i \geq \mu^\text{extd}_i \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.71)
\]

\[
\alpha^\text{prest}_i \geq \mu^\text{prest}_i \quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.72)
\]

2.5.5. Begin of service constraints

Constraints (2.5.73) state that service, i.e. loading and/or unloading of goods, starting at time \(\text{start}_{i+1}\) in vertex \(i + 1\) will exactly begin after the end of the preceding service time in vertex \(i\) (\(\Delta^\text{service}_{i}\)) plus the driving time needed to reach destination \(i + 1\) from \(i\) (\(\Delta^\text{drive}_{(i,i+1)}\)), plus the duration of all (partial) breaks and (partial) daily rest periods taken on arc \((i, i + 1)\) and in vertex \(i + 1\), plus waiting time in \(i + 1\). If a first partial break was taken prior to the departure from \(i\), 15 minutes are subtracted from the full duration of a corresponding subsequent break thus converting it to a second partial break with a duration of 30 minutes. If a break was substituted (for the substitution see page 46) by a first partial rest period, the 45 minutes for the break are subtracted.26

---

25 The extended driving times already used since the start of the week consist of extended driving times already completed with the start of a daily rest period (\(\text{noExt}\)) and of a potential driving time extension still active when the schedule starts (if \(d\text{te} = 1\)).

26 Note that first partial breaks and first partial daily rest periods are scheduled in vertices to compensate waiting time. Similar to breaks, first partial daily rest periods also reset the driving time interval and it may be advantageous to substitute a last break on an arc with a first partial daily rest period. In that case, \(\mu^\text{prest}_{i+1} = 1\) and \(\alpha^\text{prest}_{i+1} = 1\).
\[ \text{start}_{i+1} = \text{start}_i + \bar{\Delta}_{\text{service}} + \bar{\Delta}_{\text{drive}} + 45 \alpha_{i+1} \Delta_{\text{rest}} + 45 \alpha_{i+1} + \Delta_{i+1}^{\text{wait}} + 15 \alpha_{i+1}^{\text{pbreak}} - 15 \mu_{i+1}^{\text{upbreak}} - 15 \mu_{i+1}^{\text{upbreak}} + 180 \alpha_{i+1}^{\text{prest}} - 45 \mu_{i+1}^{\text{prest}} \]

\[ \forall \ i = 0, \ldots, r - 2 \]

(2.5.73)

Vertex 0 denotes the starting position. (Partial) daily rest periods and 45 minute breaks are allowed in the first vertex and the continuation of already started (partial) daily rest periods and breaks is also considered. If a (partial) break (or daily rest period) takes place at the starting time of the schedule, parameter \( \bar{u\text{b}}t \) (or \( \bar{u\text{r}}l \), respectively) specifies its duration until then. The continuation of a (partial) break or daily rest period is not mandatory. The service time \( \bar{\Delta}_{\text{service}} \) is set to be zero, that means, in (2.5.74) \( \text{start}_0 \) denotes the start of driving from vertex 0 to vertex 1. The utilization of a first partial break or a first partial daily rest period still active is also taken into consideration, where \( hpb \) indicates if a first partial break has already been taken. If a first partial daily rest period is still active, the duration of the corresponding daily rest period is adjusted accordingly to obtain a second partial daily rest period (see Section 2.5.9 on page 58).

\[ \text{start}_0 = \Delta_0^{\text{rest}} + (45 - \min(\bar{u\text{b}}t + 15 \cdot hpb, 45)) \cdot \alpha_0^{\text{break}} + (15 - \min(\bar{u\text{b}}t, 15)) \cdot \alpha_0^{\text{pbreak}} + (180 - \min(\bar{u\text{r}}l, 180)) \cdot \alpha_0^{\text{prest}} \]

(2.5.74)

### 2.5.6. Time window constraints

We model time windows as soft constraints, i.e we penalize lateness. Thus, a solution can be found even if not all time windows can be met giving additional helpful information to the dispatcher. To guarantee that exactly one time window is chosen for each vertex \( i = 1, \ldots, r - 1 \), constraints (2.5.75) are introduced. Constraints (2.5.76) state that service in vertex \( i \) will start no earlier than the lower bound of the chosen time window.

\[ \sum_{z=0}^{\text{noTW}_{i-1}} t_{w_{iz}} = 1 \quad \forall \ i = 1, \ldots, r - 1 \]

(2.5.75)

\[ \text{start}_i \geq \sum_{z=0}^{\text{noTW}_{i-1}} TW_{iz}^{\text{begin}} t_{w_{iz}} \quad \forall \ i = 1, \ldots, r - 1 \]

(2.5.76)

### 2.5.7. Lateness constraints

Lateness in vertex \( i \) is greater than or equal to the difference between the start of loading/unloading of goods and the end of the chosen time window (see (2.5.77)). The lateness
variable $\Delta_i^{late}$ is defined to be greater than or equal to zero. Therefore, $\sum_{i=1}^{r-1} \Delta_i^{late}$ represents the total lateness that is penalized in one of the objective functions.

$$\Delta_i^{late} \geq start_i - \sum_{z=0}^{noTW_i-1} TW_{iz}^{end} tw_{iz} \quad \forall i = 1, \ldots, r - 1$$ (2.5.77)

### 2.5.8. Maximum time between two consecutive weekly rest periods

The time between the end of a weekly rest period and the start of the following weekly rest period is not allowed to exceed 144 hours (8640 minutes). Therefore, each loading or unloading activity at a customer location has to end within this time interval:

$$start_i + \Delta_i^{service} \leq 8640 - ptwr \quad \forall i = 1, \ldots, r - 1$$ (2.5.78)

where $ptwr$ denotes the time passed since the last weekly rest period at the beginning of the planning horizon.

### 2.5.9. Durations of daily rest periods

Depending on the number of daily rest periods $A_{(i,i+1)}^{rest}$ scheduled on an arc $(i, i+1)$, lower bounds on their cumulative duration $\Delta_{(i,i+1)}^{rest}$ are set up. The minimum cumulated duration is reduced by 2 hours for each reduced daily rest period on arc $(i, i+1)$ and for a second partial daily rest period taken ($l_{i, prest} = 1$). The same applies to vertices but with the difference that at most one daily rest period per vertex may be scheduled. Note that if a daily rest period has been taken on arc $(i, i+1)$, a potential first partial daily rest period that has been taken in vertex $i$ or in a vertex prior to $i$ is exhausted and may not be used in vertex $i + 1$. In vertex 0, we additionally have to consider the duration of an already started rest period ($urt > 0$) and potentially a first partial daily rest period. We add the following constraints to our model accordingly:

$$\Delta_0^{rest} \geq (660 - \min(660, urt)) a_0^{rest} - 120 \mu_0^{rest} - 120 hpr \quad \forall i = 1, \ldots, r$$ (2.5.79)

$$\Delta_i^{rest} \geq 660 a_i^{rest} - 120 \mu_i^{rest} - 120 l_{i-1}^{prest} \quad \forall i = 1, \ldots, r - 1$$ (2.5.80)

$$\Delta_{i+1}^{rest} \geq 660 a_{i+1}^{rest} - 120 \mu_{i+1}^{rest} - 120 (1 - a_{(i,i+1)}^{rest}) \quad \forall i = 0, \ldots, r - 2$$ (2.5.81)

$$\Delta_{(i,i+1)}^{rest} \geq 660 A_{(i,i+1)}^{rest} - 120 \mu_{(i,i+1)}^{rest} - 120 l_i^{prest} \quad \forall i = 0, \ldots, r - 2$$ (2.5.82)
If no daily rest period is taken, we set the corresponding duration variable $\Delta_{i}^{\text{rest}}$ or $\Delta_{(i,i+1)}^{\text{rest}}$, respectively to be equal to zero:

$$
\alpha_{i}^{\text{rest}} = 0 \Rightarrow \Delta_{i}^{\text{rest}} = 0 \quad \land \\
\alpha_{(i,i+1)}^{\text{rest}} = 0 \Rightarrow \Delta_{(i,i+1)}^{\text{rest}} = 0
$$

As an upper bound on the duration of a rest period, we choose the maximum time available between two weekly rest periods, i.e. $6 \cdot 24 \text{ h} = 144 \text{ h} = 8640 \text{ min}$ and thus obtain constraints (2.5.83) and (2.5.84).

$$
\Delta_{i}^{\text{rest}} \leq 8640 \alpha_{i}^{\text{rest}} \quad \forall i = 0, \ldots, r - 1 \\
\Delta_{(i,i+1)}^{\text{rest}} \leq 8640 \alpha_{(i,i+1)}^{\text{rest}} \quad \forall i = 0, \ldots, r - 2
$$

It may be advantageous to take daily rest periods with a duration of more than 11 hours or 9 hours, respectively. This influences the beginning of the next 24 h time interval in which a daily rest period has to be taken and may be necessary to cope with subsequent time windows.

### 2.5.10. Indicator variables for daily rest periods on arcs

The variable $\alpha_{(i,i+1)}^{\text{rest}}$ is used to indicate if at least one daily rest period is taken on arc $(i, i+1)$. Hence, the following condition has to hold for each $\alpha_{(i,i+1)}^{\text{rest}}$:

$$
A_{(i,i+1)}^{\text{rest}} > 0 \Leftrightarrow \alpha_{(i,i+1)}^{\text{rest}} = 1
$$

As the number of daily rest periods during one week is bounded from above by $8640:540=16^{27}$, constraints (2.5.85) and (2.5.86) induce these conditions.$^{28}$

$$
16 \alpha_{(i,i+1)}^{\text{rest}} \geq A_{(i,i+1)}^{\text{rest}} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.85) \\
\alpha_{(i,i+1)}^{\text{rest}} \leq A_{(i,i+1)}^{\text{rest}} \quad \forall i = 0, \ldots, r - 2 \quad (2.5.86)
$$

### 2.5.11. Indicator variables for breaks on arcs

In the following, some conditions will only hold if no breaks are taken or if at least one break is taken on an arc. Therefore, variable $\alpha_{(i,i+1)}^{\text{break}}$ is introduced to indicate if at least

---

$^{27}$ A daily rest period has a duration of at least $9 \text{ h} = 540 \text{ min}$ and the time between two weekly rest periods is at most $24 \cdot 6 \text{ h} = 144 \text{ h} = 8640 \text{ min}$.

$^{28}$ For the derivation see (f1') and (f2') on page 32.
one break is taken on arc \((i, i + 1)\) or not. We wish to state that

\[ A_{(i,i+1)}^{\text{break}} > 0 \Leftrightarrow \alpha_{(i,i+1)}^{\text{break}} = 1. \]

An upper bound on the maximum number of breaks during one week is \(8640 : 45 = 192\). Again, using a big-M approach, the above statement is expressed by (2.5.87) and (2.5.88).

\[
\begin{align*}
192 \alpha_{(i,i+1)}^{\text{break}} &\geq A_{(i,i+1)}^{\text{break}} & \forall \ i = 0, \ldots, r - 2 & \quad (2.5.87) \\
\alpha_{(i,i+1)}^{\text{break}} &\leq A_{(i,i+1)}^{\text{break}} & \forall \ i = 0, \ldots, r - 2 & \quad (2.5.88)
\end{align*}
\]

## 2.5.12. Decision variables that indicate a necessary break

The binary variable \(l_{i}^{\text{bn}}\) indicates whether a break is necessary to completely use the daily driving time left when leaving vertex \(i\) (\(L_{i}^{\text{ddt}}\)) or not. This is the case if \(L_{i}^{\text{ddt}} > L_{i}^{\text{dt}}\), where \(L_{i}^{\text{dt}}\) denotes the driving time left until the next break:

\[ L_{i}^{\text{ddt}} > L_{i}^{\text{dt}} \Leftrightarrow l_{i}^{\text{bn}} = 1 \]

This is imposed by the following constraints:

\[
\begin{align*}
270 \ l_{i}^{\text{bn}} &\geq L_{i}^{\text{ddt}} - L_{i}^{\text{dt}} & \forall \ i = 0, \ldots, r - 1 & \quad (2.5.89) \\
\ l_{i}^{\text{bn}} &\leq L_{i}^{\text{ddt}} - L_{i}^{\text{dt}} & \forall \ i = 0, \ldots, r - 1 & \quad (2.5.90)
\end{align*}
\]

Note that (2.5.90) induces \(L_{i}^{\text{ddt}} \geq L_{i}^{\text{dt}}\) and (2.5.89) ensures that the maximum difference between \(L_{i}^{\text{ddt}}\) and \(L_{i}^{\text{dt}}\) is less than or equal to 270 min = 4.5h. \(L_{i}^{\text{ddt}} - L_{i}^{\text{dt}}\) is always either equal to zero, or greater than or equal to one, non-integer values between 0 and 1 are not possible due to the above constraints.

## 2.5.13. Decision variables that indicate that a break has already been taken

The binary variable \(e_{i}^{\text{bt}}\) indicates if the daily driving time left when entering vertex \(i\), \(E_{i}^{\text{ddt}}\), is greater than the driving time left until the next break has to be taken, \(E_{i}^{\text{dt}}\), i.e. we wish to achieve that

\[ E_{i}^{\text{ddt}} > E_{i}^{\text{dt}} \Leftrightarrow e_{i}^{\text{bt}} = 0. \]
This represented by constraints (2.5.91) and (2.5.92).

\begin{align*}
2.5.14. \text{Indicator variables for early daily rest periods} \\
\alpha_{\text{rest}}(i,i+1) = 0 & \Rightarrow \mu_{\text{earlydr}1}(i,i+1) = 0 \\
A_{\text{rest}}(i,i+1) < 2 & \Rightarrow \mu_{\text{earlydr}2}(i,i+1) = 0
\end{align*}
2. Scheduling of driving times, breaks and rest periods

\[ I_{d}^{dt} = 180 \]
\[ I_{d}^{ddt} = 450 \]
\[ L_{0} = 480 \]

\[ [17:30, 18:30] \quad [20:00, 23:00] \]

\[ \Delta_{drive}^{(0,1)} = 420 \]
\[ \Delta_{service} = 60 \]
\[ \Delta_{drive}^{(1,2)} = 180 \]

\[ \text{start in 0 at time: } 00:00 \]

\[ \text{without early daily rest:} \]

<table>
<thead>
<tr>
<th>duration</th>
<th>end of activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>drive</td>
<td>03:00</td>
</tr>
<tr>
<td>break</td>
<td>00:45</td>
</tr>
<tr>
<td>drive</td>
<td>04:00</td>
</tr>
<tr>
<td>rest</td>
<td>11:00</td>
</tr>
</tbody>
</table>

\[ \text{start service in 1:} 18:45 \]

<table>
<thead>
<tr>
<th>lateness</th>
<th>00:15</th>
</tr>
</thead>
</table>

\[ \text{service} \quad 01:00 \quad 19:45 \]
\[ \text{drive} \quad 03:00 \quad 22:45 \]
\[ \text{arrival in 2} \quad 22:45 \]

\[ \text{with early daily rest:} \]

<table>
<thead>
<tr>
<th>duration</th>
<th>end of activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>drive</td>
<td>03:00</td>
</tr>
<tr>
<td>rest</td>
<td>11:00</td>
</tr>
<tr>
<td>drive</td>
<td>04:00</td>
</tr>
</tbody>
</table>

\[ \text{start service in 1:} 18:00 \]

<table>
<thead>
<tr>
<th>lateness</th>
<th>00:00</th>
</tr>
</thead>
</table>

\[ \text{service} \quad 01:00 \quad 19:00 \]
\[ \text{drive} \quad 00:30 \quad 19:30 \]
\[ \text{break} \quad 00:45 \quad 20:15 \]
\[ \text{drive} \quad 02:30 \quad 22:45 \]
\[ \text{arrival in 2} \quad 22:45 \]

Figure 2.9.: The impact of an early daily rest period \( \mu_{(1,2)}^{early} = 1 \)

\[ \mu_{(i,i+1)}^{early} \leq \alpha_{rest}^{(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.93) \]

\[ \mu_{(i,i+1)}^{early} \leq \Delta_{rest}^{(i,i+1)} - \alpha_{rest}^{(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.94) \]

If \( L_{d}^{ddt} = L_{d}^{dt} \) then \( I_{b}^{ln} = 0 \) (because of (2.5.89) and (2.5.90)), this means that no break is necessary to completely use \( L_{d}^{ddt} \). In this case, the subsequent daily rest period on arc \( (i, i + 1) \) (if there has to be one) has to be a regular daily rest period. No early daily rest period of type 1 can be scheduled. Therefore,

\[ I_{b}^{ln} = 0 \Rightarrow \mu_{(i,i+1)}^{early} = 0 \]

We obtain the following constraints:

\[ \mu_{(i,i+1)}^{early} \leq I_{b}^{ln} \quad \forall \ i = 0, \ldots, r - 2 \quad (2.5.95) \]
2.5.15. Vertex activity constraints

The vertex activity constraints generally limit possible activities and their combinations in a vertex.

In vertex 0, we decide to set waiting time to be zero (see (2.5.96)), as there is no time window:

\[ \Delta_{\text{wait}}^0 = 0 \]  \hspace{1cm} (2.5.96)

Moreover, solutions with waiting time \( \Delta_{\text{wait}}^0 \) greater than zero would be equivalent (w.r.t. the objective function value) to solutions with waiting time in the following vertex in case no daily rest periods on arc \((0, 1)\) were scheduled. If a daily rest period is taken on this arc, it may be extended accordingly.

If a break or daily rest period (full or partial) has not started yet, it may be postponed to the following arc \((0, 1)\). Thus, to reduce the solution space, we add the constraints

\[
\begin{align*}
\alpha_{\text{break}}^0 &= 0 \text{ and } \alpha_{\text{break}}^0 = 0 \text{ if } \bar{\text{ubt}} = 0 \\
\alpha_{\text{rest}}^0 &= 0 \text{ and } \alpha_{\text{prest}}^0 = 0 \text{ if } \bar{\text{urt}} = 0
\end{align*}
\]  \hspace{1cm} (2.5.97)

\hspace{1cm} (2.5.98)

and thereby prohibit starting a new rest period or break.

There is no clear rule concerning the time between a daily rest period and a partial break or partial rest. We assume that it is not desired by the legislator that these resting activities are scheduled directly in series and only allow one resting activity per vertex:

\[
\alpha_i^{\text{rest}} + \alpha_i^{\text{break}} + \alpha_i^{\text{prest}} + \alpha_i^{\text{prest}} - \mu_i^{\text{prest}} \leq 1 \quad \forall \ i = 0, \ldots, r - 1
\]  \hspace{1cm} (2.5.99)

If a break is substituted by a partial daily rest period \( (\alpha_i^{\text{prest}} = 1 \land \mu_i^{\text{prest}} = 1) \) on arc \((i - 1, i)\), then a partial break for a driving time extension on a subsequent arc may be advantageous in contrast to the resting activities \( \alpha_i^{\text{break}} = 1 \) or \( \alpha_i^{\text{rest}} = 1 \). We add constraints (2.5.100) to further reduce the solution space.

\[
\alpha_i^{\text{rest}} + \alpha_i^{\text{break}} + \alpha_i^{\text{prest}} \leq 1 \quad \forall \ i = 0, \ldots, r - 1
\]  \hspace{1cm} (2.5.100)

2.5.16. Get status constraints

The driver starts at vertex 0 with a certain status, which depends on former activities.\(^{29}\) The time left until the next daily rest period \( (E_0^i) \) depends on the time elapsed since the end of the last daily rest period. The daily driving time left until the next daily rest period

\(^{29}\) See page 27 for a short description of the status variables.
(\(E^{ddt}_0\)) depends on the current daily driving time and the time left until the next daily rest period. The driving time left until the next break or daily rest period (\(E^{ddt}_0\)) depends on the uninterrupted current driving time, the overall time spent driving since the last daily rest period and the time left until the next daily rest period. In addition, each of the status variables is influenced by a decision about a reduced daily rest period (\(\mu_0^{dredrest} = 1\)) and by a daily rest period scheduled in vertex 0.

The time left until the next daily rest period has to be taken (\(E^t_0\)) depends on the time spent since the last daily rest period \(ptr\) and if a first partial rest was already made (\(hpr = 1\)) in case no reduced daily rest period is taken in vertex 0.

As it may be the case that \(ptr\) is larger than actually allowed, \(780 - ptr + 120 hpr\) may be less than zero. The following logical condition takes this into account:

\[
\mu_0^{dredrest} = 0 \Rightarrow E^t_0 = \max\{780 - ptr + 120 hpr, 0\}
\]

For our model, we obtain constraints (2.5.101) and (2.5.102).

\[
\begin{align*}
E^t_0 &\leq \max\{780 - ptr + 120 hpr, 0\} + 120 \mu_0^{dredrest} & (2.5.101) \\
E^t_0 &\geq \max\{780 - ptr + 120 hpr, 0\} & (2.5.102)
\end{align*}
\]

If a reduced daily rest period is planned after leaving vertex 0 (\(\mu_0^{dredrest} = 1\)), we have to differentiate between two cases. In case one, no daily rest period is taken in vertex 0. In that case, two hours are added to \(E^t_0\) to be able to already use the additional time for activities in vertex 0. In case two, a daily rest period is taken in vertex 0. In that case, two hours are added to \(L^t_0\) as the time left until the next rest period is reset by the daily rest period taken. The following two conditions must hold:

\[
(\mu_0^{dredrest} = 1 \land \alpha_0^{rest} = 0) \Rightarrow E^t_0 = \max\{900 - ptr, 0\}
\]
\[
(\mu_0^{dredrest} = 1 \land \alpha_0^{rest} = 1) \Rightarrow E^t_0 = \max\{780 - ptr, 0\}
\]

The following constraints enforce the above conditions:

\[
\begin{align*}
E^t_0 &\leq \max\{780 - ptr + 120 hpr, 0\} + 120 (1 - \alpha_0^{rest}) & (2.5.103) \\
E^t_0 &\leq \max\{900 - ptr, 0\} & (2.5.104) \\
E^t_0 &\geq \max\{900 - ptr, 0\} - 120 (1 - \mu_0^{dredrest}) - 120 \alpha_0^{rest} & (2.5.105)
\end{align*}
\]

Note that a reduced daily rest period will not be taken if a second partial rest can be made instead (see constraints (2.5.46) on page 50).

Now, the daily driving time left until the next daily rest period (\(E^{ddt}_0\)) can be determined. \(E^{ddt}_0\) depends on the driving time already used since the last daily rest period and a potential driving time extension. In addition, \(E^{ddt}_0\) is bounded from above by the time left until the
2.5. Mathematical formulation

next daily rest period $E^t_0$. We obtain the following conditions:

$$(\mu^{dredrest}_0 = 0 \lor \alpha^{rest}_0 = 1)$$

$$\Rightarrow E^{ddt}_0 = \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\}$$

$$(\mu^{dredrest}_0 = 1 \land \alpha^{rest}_0 = 0)$$

$$\Rightarrow E^{ddt}_0 = \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\}$$

By using the big-M approach and considering the fact that the different cases may only differ by at most 120 minutes as far as the decision about a reduced daily rest period $\mu^{dredrest}_0$ is concerned, we add the following constraints to our model:

$$E^{ddt}_0 \leq \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\} + 120 \mu^{dredrest}_0 \quad (2.5.106)$$

$$E^{ddt}_0 \geq \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\} \quad (2.5.107)$$

$$E^{ddt}_0 \leq \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\} + 120 (1 - \alpha^{rest}_0) \quad (2.5.108)$$

$$E^{ddt}_0 \leq \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 900 - \overline{ptr}\}, 0\} \quad (2.5.109)$$

$$E^{ddt}_0 \geq \max\{\min\{540 + 60 \overline{dt} - \overline{ddt}, 900 - \overline{ptr}\}, 0\} - 120 (1 - \mu^{dredrest}_0) - 120 \alpha^{rest}_0 \quad (2.5.110)$$

The last continuous status variable to be determined for vertex $0$ is $E^{dt}_0$. In addition to the dependencies on $E^t_0$ and $E^{ddt}_0$, the driving time since the last break or rest period, $\overline{udt}$, needs to be considered:

$$(\mu^{dredrest}_0 = 0 \lor \alpha^{rest}_0 = 1)$$

$$\Rightarrow E^{dt}_0 = \max\{\min\{270 - \overline{udt}, 540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\}$$

$$(\mu^{dredrest}_0 = 1 \land \alpha^{rest}_0 = 0)$$

$$\Rightarrow E^{dt}_0 = \max\{\min\{270 - \overline{udt}, 540 + 60 \overline{dt} - \overline{ddt}, 900 - \overline{ptr}\}, 0\}$$

We add the following constraints to our model:

$$E^{dt}_0 \leq \max\{\min\{270 - \overline{udt}, 540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\} + 120 \mu^{dredrest}_0 \quad (2.5.111)$$

$$E^{dt}_0 \geq \max\{\min\{270 - \overline{udt}, 540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\} \quad (2.5.112)$$

$$E^{dt}_0 \leq \max\{\min\{270 - \overline{udt}, 540 + 60 \overline{dt} - \overline{ddt}, 780 - \overline{ptr} + 120 \overline{hpr}\}, 0\} + 120 (1 - \alpha^{rest}_0) \quad (2.5.113)$$

$$E^{dt}_0 \leq \max\{\min\{270 - \overline{udt}, 540 + 60 \overline{dt} - \overline{ddt}, 900 - \overline{ptr}\}, 0\} \quad (2.5.114)$$
\[ E_0^{dt} \geq \max \{ \min \{ 270 - \overline{udt}, 540 + 60 \overline{dtc} - \overline{ddt}, 900 - \overline{ptr} \}, 0 \} \\
- 120(1 - \mu_0^{dredrest}) - 120 \alpha_0^{rest} \tag{2.5.115} \]

### 2.5.17. Continuous driver status variables when entering a vertex

The driver status variables when entering vertex \( i + 1 \) \( (E_{i+1}^{dt}, E_{i+1}^{ddt} \) and \( E_{i+1}^{l} \)) are determined based on the driver status variables when leaving vertex \( i \) and the driver activities on arc \((i, i + 1)\). Driver activities on arcs include resting, taking a break and driving. If a first partial daily rest period or break has been taken in a preceding vertex, second partial rests and breaks can be scheduled. The interrelation of status variables and activities on arcs is shown in Figure 2.5 on page 26.

The constraints in this section serve two purposes. One is to determine the necessary number, duration and timing of rest periods and breaks for traversing an arc \((i, i + 1)\) depending on the driver status \( L_i^{dt}, L_i^{ddt} \) and \( L_i^{l} \) when leaving \( i \). The second purpose is the determination of the resulting driver status, \( E_i^{dt}, E_i^{ddt} \) and \( E_i^{l} \) when entering vertex \( i + 1 \).

We introduce variable \( \lambda_i^5 \) to indicate if a driving time extension of one complete hour before the first daily rest period (if there is one) on arc \((i, i + 1)\) is possible, without exceeding the time left until the next rest period \((L_i^l)\). In case no driving time extension is considered on arc \((i, i + 1)\) before the first daily rest period, \( \lambda_i^5 \) is set to be equal to 1.

\[
(\mu_{extd}^{(i,i+1)}) = 1 \land L_i^l > L_i^{ddt} + 60 + 45 p_{ln} + 45 - 15 \mu_{upbreak}^{(i,i+1)} ) \Rightarrow \lambda_i^5 = 1 \\
(\mu_{extd}^{(i,i+1)}) = 1 \land L_i^l < L_i^{ddt} + 60 + 45 p_{ln} + 45 - 15 \mu_{upbreak}^{(i,i+1)} ) \Rightarrow \lambda_i^5 = 0 \\
\mu_{extd}^{(i,i+1)} = 0 \Rightarrow \lambda_i^5 = 1
\]

In case that \( L_i^l = L_i^{ddt} + 60 + 45 p_{ln} + 45 - 15 \mu_{upbreak}^{(i,i+1)} \) and \( \mu_{extd}^{(i,i+1)} = 1 \), \( \lambda_i^5 \) will not be uniquely determined. This causes no problem as the corresponding constraints in that case will yield the same variable values for \( E_i^{dt} \) or \( E_i^{ddt} \), no matter if \( \lambda_i^5 = 1 \) or \( \lambda_i^5 = 0 \).\(^{30}\)

We use the big-M approach to represent the above statements.

\[
270 \lambda_i^5 \geq L_i^l - L_i^{ddt} - 60 - 45 l_{ln} - 45 + 15 \mu_{upbreak}^{(i,i+1)} \\
- 270 (1 - \mu_{extd}^{(i,i+1)}) \quad \forall \ i = 0, \ldots, r - 2 \tag{2.5.116} \\
150 (\lambda_i^5 - 1) \leq L_i^l - L_i^{ddt} - 60 - 45 l_{ln} - 45 + 15 \mu_{upbreak}^{(i,i+1)} \\
+ 150 (1 - \mu_{extd}^{(i,i+1)}) \quad \forall \ i = 0, \ldots, r - 2 \tag{2.5.117} \\
\lambda_i^5 \geq 1 - \mu_{extd}^{(i,i+1)} \quad \forall \ i = 0, \ldots, r - 2 \tag{2.5.118}
\]

\(^{30}\) See Section 2.5.1, page 31 for a more detailed description.
Because of having a maximum difference of \(900 - 540 = 360\) minutes for \(L_i - \ddt_i\), we set \(M = 360 - 60 - 45 + 15 = 270\) in inequalities \((2.5.116)\). Knowing that \(L_i \geq \ddt_i\), we choose \(M = 60 + 45 + 45 = 150\) in constraints \((2.5.117)\).

**Driving time left until the next break or rest**

We will now determine \(E_{i+1}^{dt}\), the driving time left until the next break or daily rest period when entering vertex \(i\). The value of this variable depends on the driver status when leaving the preceding vertex \(i\) and the driver activities scheduled for arc \((i, i+1)\).

We have to differentiate between different cases when setting the value of \(E_{i+1}^{dt}\) depending on

- the driver status \(L_i^{dt}\) when leaving the previous vertex
- if an early daily rest period is taken as the first resting activity on arc \((i, i+1)\),
- the value of \(\lambda^5_i\)
- if a break is scheduled on the arc \((i, i+1)\).

Breaks and daily rest periods scheduled on the arc \((i, i+1)\) "extend" the driving time allowed until the next break or daily rest period starting with the value of the status variable \(E_i^{dt}\). In most instances, each rest period or break allows 4.5 hours of additional driving to traverse the arc \((i, i+1)\). An exception can be the first resting activity on an arc if it is a break. The time left until the next daily rest period \(L_i\) may, due to waiting, loading and unloading activities in the past, not suffice to schedule another 4.5 hours of driving after this break. Another exception is a break that is taken for daily driving time extension which may extend the daily driving time by one hour (from 9 to a maximum of 10 hours). If such a break is made before the first daily rest period on that arc, it may be the case that the extension is less than 60 minutes as the maximum daily driving time is bounded from above by the maximum time between two consecutive daily rest periods.

Let us first determine the driving time left if no early daily rest period is taken as first resting activity \((\mu_{(i,i+1)}^{early dr} = 0)\) and either a complete driving time extension of 60 minutes is possible before the first daily rest period on arc \((i, i+1)\) or such a driving time extension is not made \((\lambda^5_i = 1)\). In addition, we demand that at least one break is necessary to traverse arc \((i, i+1)\).

\[
\begin{align*}
\mu_{(i,i+1)}^{early dr} = 0 & \land \lambda^5_i = 1 \land \alpha^\text{break}_i = 1 \\
\Rightarrow E_i^{dt} &= r_i^{dt} - 270 A_{i,i+1}^{\text{break}} - 270 b_{i} - 270 A_{i,i+1}^{\text{rest}} - \Delta_{i,i+1}^{\text{drive}} \\
&\quad - 210 \mu_{(i,i+1)}^{\text{extd}1} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3}
\end{align*}
\]

\[\tag{31}\]
Decision variables for early daily rest periods: see \((2.5.93) - (2.5.94)\) on page 62.
If the first resting activity on arc \((i, i+1)\) is a break \(\alpha_{(i,i+1)}^{\text{break}} = 1 \land t_{(i,i+1)}^{\text{break}} = 1 \land \mu_{(i,i+1)}^{\text{earlydr}} = 0\), it may be the case that \(L_i^{\text{ddt}} < L_i^{\text{dt}} + 270\), as \(L_i^{\text{ddt}}\) is bounded from above by the time left until the next daily rest period \(L_i^{\text{r}}\), which is influenced by former servicing and waiting times.\footnote{Another reason may be that a break in a preceding vertex was taken after less than 4.5 hours of driving.} The first break then serves to completely use \(L_i^{\text{ddt}}\), i.e. it extends the driving time by \(L_i^{\text{ddt}} - L_i^{\text{dt}}\). Therefore, 270 minutes \(270 t^{\text{bn}}_{i,i+1}\) need to be subtracted. In addition, 210 minutes need to be subtracted for each extended daily driving time, as the corresponding breaks only extend driving by 60, not by 270 minutes.

If a daily driving time extension \(\mu_{(i,i+1)}^{\text{extd}} = 1\) is used and \(\lambda_5 = 0\), then the remaining daily driving time depends on the time left until the next daily rest period is necessary \(L_i^{\text{r}}\) minus 45 minutes for the break needed for the driving time extension minus 45 minutes if \(L_i^{\text{ddt}} > L_i^{\text{dt}} (t_{i,i+1}^{\text{bn}} = 1)\) plus 15 minutes if a partial break has already been taken.

\[
\lambda_5^5 = 0 \\
\Rightarrow E_{i+1}^{\text{dt}} = L_i^{\text{dt}} + 270 A_{(i,i+1)}^{\text{break}} - 270 - 45 - 270 t_{i,i+1}^{\text{bn}} - 45 t_{i,i+1}^{\text{bn}} + 15 \mu_{(i,i+1)}^{\text{upbreak}} + 270 A_{(i,i+1)}^{\text{rest}} - \bar{\Delta}_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3}
\]

If an early daily rest period is scheduled (i.e. \(\mu_{(i,i+1)}^{\text{earlydr}} = 1\)), the first resting activity on arc \((i, i+1)\) is a daily rest period which "extends" the driving time until the next break or rest period \(L_i^{\text{dt}}\) by 4.5 hours.

\[
\Rightarrow E_{i+1}^{\text{dt}} = L_i^{\text{dt}} + 270 A_{(i,i+1)}^{\text{break}} + 270 A_{(i,i+1)}^{\text{rest}} - \bar{\Delta}_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3}
\]

For the case that no break is made on arc \((i, i+1)\), we simply obtain the logical condition

\[
\alpha_{(i,i+1)}^{\text{break}} = 0 \Rightarrow E_{i+1}^{\text{dt}} = L_i^{\text{dt}} + 270 A_{(i,i+1)}^{\text{rest}} - \bar{\Delta}_{(i,i+1)}^{\text{drive}}
\]

Reformulating the above conditions by using the big-M approach and integrating upper and lower bounds we obtain the following linear constraints:

\[
E_{i+1} \leq L_i^{\text{ddt}} + 270 A_{(i,i+1)}^{\text{break}} - 270 t_{i,i+1}^{\text{bn}} + 270 A_{(i,i+1)}^{\text{rest}} - \bar{\Delta}_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}1} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3} + 480 \mu_{(i,i+1)}^{\text{earlydr}} + 480 (1 - \alpha_{(i,i+1)}^{\text{break}}) + 480 (1 - \lambda_5^5) \\
\forall i = 0, \ldots, r - 2 \tag{2.5.119}
\]

\[
E_{i+1} \geq L_i^{\text{ddt}} + 270 A_{(i,i+1)}^{\text{break}} - 270 t_{i,i+1}^{\text{bn}} + 270 A_{(i,i+1)}^{\text{rest}} - \bar{\Delta}_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}1} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3} - 630 (1 - \lambda_5^5)
\]
∀ i = 0, . . . , r − 2
\[ E_{i+1}^{dt} \leq L_i + 270 A_{\text{break}}^d(i,i+1) - 270 - 45 - 270 l_{i}^{bm} - 45 l_{i}^{lm} + 15 \mu_{(i,i+1)}^{\text{upbreak}} + 270 A_{\text{est}}^d(i,i+1) - \Delta_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3} + 630 \lambda_5^i \] \forall i = 0, . . . , r − 2 \tag{2.5.120}

∀ i = 0, . . . , r − 2
\[ E_{i+1}^{dt} \geq L_i + 270 A_{\text{break}}^d(i,i+1) - 270 - 45 - 270 l_{i}^{bm} - 45 l_{i}^{lm} + 15 \mu_{(i,i+1)}^{\text{upbreak}} + 270 A_{\text{est}}^d(i,i+1) - \Delta_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3} - 270 \lambda_5^i \] \forall i = 0, . . . , r − 2 \tag{2.5.121}

∀ i = 0, . . . , r − 2
\[ E_{i+1}^{dt} \leq L_i + 270 A_{\text{break}}^d(i,i+1) + 270 A_{\text{est}}^d(i,i+1) - \Delta_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3} \] \forall i = 0, . . . , r − 2 \tag{2.5.122}

∀ i = 0, . . . , r − 2
\[ E_{i+1}^{dt} \geq L_i + 270 A_{\text{break}}^d(i,i+1) + 270 A_{\text{est}}^d(i,i+1) - \Delta_{(i,i+1)}^{\text{drive}} - 210 \mu_{(i,i+1)}^{\text{extd}2} - 210 \mu_{(i,i+1)}^{\text{extd}3} - 1110 (1 - \mu_{(i,i+1)}^{\text{earlydr}1}) \] \forall i = 0, . . . , r − 2 \tag{2.5.123}

∀ i = 0, . . . , r − 2
\[ E_{i+1}^{dt} \geq L_i + 270 A_{\text{est}}^d(i,i+1) - \Delta_{(i,i+1)}^{\text{drive}} \] \forall i = 0, . . . , r − 2 \tag{2.5.124}

Constraints (2.5.123) set an upper bound for \( E_{i+1}^{dt} \). Therefore, no big-M terms are needed for these constraints. Lower bounds on \( E_{i+1}^{dt} \) are imposed by constraints (2.5.120) in case \( \lambda_5^i = 1 \) and by constraints (2.5.122) if \( \lambda_5^i = 0 \). Accordingly, \( M \) is chosen for the remaining constraints such that they become redundant if the corresponding conditions do not hold. Often, the following dependencies are used:

- \( L_i^{dt} \leq L_i^{ddt} \leq L_i^{d} \) \∀ i = 0, . . . , r − 1
- \( L_i^{d} \leq L_i^{ddt} + 360 \) \∀ i = 0, . . . , r − 1

Big-M terms are always shown on the right-hand side of the constraint and appear at the end.

Finally, we demand that no break is made if it is not necessary. That means, if \( L_i^{dt} > \Delta_{(i,i+1)}^{\text{drive}} \), then no break is scheduled.

\[ L_i^{dt} > \Delta_{(i,i+1)}^{\text{drive}} \Rightarrow \alpha_{(i,i+1)}^{\text{break}} = 0 \]

This is guaranteed by

\[ 270 (1 - \alpha_{(i,i+1)}^{\text{break}}) \geq L_i^{dt} - \Delta_{(i,i+1)}^{\text{drive}} \] \∀ i = 0, . . . , r − 2 \tag{2.5.126}
Daily driving time left

Now, the driving time left until the next daily rest period when entering vertex \( i \), \( E^\text{ddt}_i \), will be determined. Similarly as in the last section, rest periods are considered to "extend" the driving time allowed until the next daily rest period starting with the value of the status variable \( E^\text{ddt}_i \). Each daily rest period allows 9 hours of additional driving to traverse the arc \(( i, i+1)\). Each extended daily driving time allows one additional hour of driving, except for the case that the daily driving time is extended before the first daily rest period is taken (if one is taken) on arc \(( i, i+1)\), as \( L^t_i \), the time left until the next daily rest period when leaving vertex \( i \) may not suffice to schedule another 60 minutes of driving. An early daily rest period as first resting activity \( \mu_\text{earlydr}^1( i, i+1) = 1 \) reduces the driving time until the next daily rest period from \( L^t_i \) to \( L^\text{dt}_i \), an early daily rest period as last resting activity reduces the driving time between the last two daily rest periods on arc \(( i, i+1)\) from 9 to 4.5 hours.

At first, let us set the logical condition for the case in which a driving time extension before the first daily rest period (if one is scheduled) is possible, without exceeding \( L^t_i \) \(( \lambda^5_i = 1)\) and no early daily rest period as first resting activity is scheduled \(( \mu_\text{earlydr}^1( i, i+1) = 0)\):

\[
(\mu_\text{earlydr}^1( i, i+1) = 0 \land \lambda^5_i = 1) \\
\Rightarrow E^\text{ddt}_{i+1} = L^t_i + 540 A^\text{rest}_{(i,i+1)} - 270 \mu_\text{earlydr}^2( i, i+1) - \bar{\Delta}_{\text{drive}}( i,i+1) \\
+ 60 \mu^\text{extd1}_{(i,i+1)} + 60 \mu^\text{extd2}_{(i,i+1)} + 60 \mu^\text{extd3}_{(i,i+1)}
\]

The next condition considers the case that a driving time extension is scheduled before the first daily rest period, but the time \( L^t_i \) does not suffice to schedule another 60 minutes of driving.\(^{33}\)

\[
\lambda^5_i = 0 \\
\Rightarrow E^\text{ddt}_{i+1} = L^t_i - 45 l^\text{bn}_i - 45 + 15 \mu^\text{upbreak}_{(i,i+1)} + 540 A^\text{rest}_{(i,i+1)} - 270 \mu_\text{earlydr}^2( i, i+1) - \bar{\Delta}_{\text{drive}}( i,i+1) \\
+ 60 \mu^\text{extd2}_{(i,i+1)} + 60 \mu^\text{extd3}_{(i,i+1)}
\]

In case an early daily rest period is the first resting activity on arc \(( i, i+1) \) \(( \mu_\text{earlydr}^1( i, i+1) = 1)\), a driving time extension before the first daily rest period is not possible. We obtain the logical condition

\[
\mu_\text{earlydr}^1( i, i+1) = 1 \\
\Rightarrow E^\text{ddt}_{i+1} = L^\text{dt}_i + 540 A^\text{rest}_{(i,i+1)} - 270 \mu_\text{earlydr}^2( i, i+1) - \bar{\Delta}_{\text{drive}}( i,i+1) + 60 \mu^\text{extd2}_{(i,i+1)} + 60 \mu^\text{extd3}_{(i,i+1)}
\]

Again, we reformulate the above conditions by using the big-M approach, integrate upper

\(^{33}\) Note that in this case \( \mu_\text{earlydr}^1( i, i+1) = 0 \) because of (2.5.118) on page 66 and (2.5.54) on page 53.
and lower bounds, and obtain the following linear constraints:

\[
\begin{align*}
E_i^{ddt} &\leq L_i^{ddt} + 540 \ A_{rest}^{i,(i+1)} - 270 \ \mu_{\text{earlydr}^2} - \Delta_{\text{drive}}^{i,(i+1)} \\
&\quad + 60 \ \mu_{\text{extd}^1} + 60 \ \mu_{\text{extd}^2} + 60 \ \mu_{\text{extd}^3} \\
&\quad \forall \ i = 0, \ldots , r - 2 \\
E_i^{ddt} &\geq L_i^{ddt} + 540 \ A_{rest}^{i,(i+1)} - 270 \ \mu_{\text{earlydr}^2} - \Delta_{\text{drive}}^{i,(i+1)} \\
&\quad + 60 \ \mu_{\text{extd}^1} + 60 \ \mu_{\text{extd}^2} + 60 \ \mu_{\text{extd}^3} \\
&\quad - 330 \ \mu_{\text{earlydr}^1} - 330 \ (1 - \lambda_i^5) \\
&\quad \forall \ i = 0, \ldots , r - 2 \\
E_i^{ddt} &\leq L_i^t - 45 \ \mu_{\text{fn}}^i - 45 + 15 \ \mu_{\text{upbreak}} + 540 \ A_{rest}^{i,(i+1)} - 270 \ \mu_{\text{earlydr}^2} - \Delta_{\text{drive}}^{i,(i+1)} \\
&\quad + 60 \ \mu_{\text{extd}^2} + 60 \ \mu_{\text{extd}^3} \\
&\quad + 150 \ \lambda_i^5 \\
&\quad \forall \ i = 0, \ldots , r - 2 \\
E_i^{ddt} &\geq L_i^t - 45 \ \mu_{\text{fn}}^i - 45 + 15 \ \mu_{\text{upbreak}} + 540 \ A_{rest}^{i,(i+1)} - 270 \ \mu_{\text{earlydr}^2} - \Delta_{\text{drive}}^{i,(i+1)} \\
&\quad + 60 \ \mu_{\text{extd}^2} + 60 \ \mu_{\text{extd}^3} \\
&\quad - 600 \ \lambda_i^5 \\
&\quad \forall \ i = 0, \ldots , r - 2 \\
E_i^{ddt} &\leq L_i^t + 540 \ A_{rest}^{i,(i+1)} - 270 \ \mu_{\text{earlydr}^2} - \Delta_{\text{drive}}^{i,(i+1)} \\
&\quad + 60 \ \mu_{\text{extd}^2} + 60 \ \mu_{\text{extd}^3} \\
&\quad + 330 \ (1 - \mu_{\text{earlydr}^1}) \\
&\quad \forall \ i = 0, \ldots , r - 2 \\
E_i^{ddt} &\geq L_i^t + 540 \ A_{rest}^{i,(i+1)} - 270 \ \mu_{\text{earlydr}^2} - \Delta_{\text{drive}}^{i,(i+1)} \\
&\quad + 60 \ \mu_{\text{extd}^2} + 60 \ \mu_{\text{extd}^3} \\
&\quad \forall \ i = 0, \ldots , r - 2 \\
\end{align*}
\]  

Constraints (2.5.127) and (2.5.132) impose upper and lower bounds on \(E_i^{ddt}\), respectively. Therefore, no big-M terms are needed for these constraints. Accordingly, M is chosen for the remaining constraints such that they become redundant when the corresponding conditions do not hold. The following bounds are used:

\begin{itemize}
  \item \(L_i^{dt} \leq L_i^{ddt} \leq L_i^t\) \quad \forall \ i = 0, \ldots , r - 1
  \item \(L_i^t \leq L_i^{dt} + 630\) \quad \forall \ i = 0, \ldots , r - 1
  \item \(L_i^{ddt} \leq L_i^{dt} + 270\) \quad \forall \ i = 0, \ldots , r - 1 \text{ (see (2.5.89) on page 60)}
\end{itemize}
Additionally, we demand that the driving time left until the next daily rest period or break is less than or equal to the daily driving time left:

\[ E_i^d t \leq E_i^{dd t} \quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.133) \]

**Maximum time until the next daily rest period**

When traveling long distances, often, reaching the maximum daily driving time allowed forces a new daily rest period and the time between two daily rest periods is less than 24 - 11 = 13 or 24 - 9 = 15 hours, respectively.

The time left until the next daily rest period when entering vertex \( i \) (i.e. \( E_i^t \)) can be derived from the daily driving time left until the next rest period \( E_i^{dd t} \) in case at least one daily rest period was taken on arc \((i, i + 1)\). Otherwise, \( E_i^t \) depends on the status when leaving the preceding vertex \( i \) and the overall duration of activities when traversing the arc \((i, i + 1)\).

In addition, we have to consider if a daily rest period is taken in vertex \( i + 1 \). If the next daily rest period after leaving vertex \( i + 1 \) is planned to be a reduced one \((\mu_i^{dredrest} = 1)\) and a daily rest period is taken in \( i + 1 \), this will only affect \( L_{i+1} \), the time left until the next daily rest period when leaving vertex \( i + 1 \). If no daily rest period is taken in \( i + 1 \) and a daily rest period was taken on arc \((i, i + 1)\), \( E_i^t \) needs to be modified in case that \( \mu_i^{dredrest} = 1 \).

Let us first consider the case that at least one daily rest period is taken on arc \((i, i + 1)\) and no daily rest period is scheduled for vertex \( i + 1 \). The time left until the next daily rest period when entering vertex \( i + 1 \) exceeds the daily driving time left until the next period by \( 13 \text{ h} - 9 \text{ h} = 240 \text{ min} \), minus 45 minutes if a break has already been taken since the last daily rest period on arc \((i, i + 1)\) \((e_{bt,i+1} = 1)\) plus \( 2 \text{ h} = 120 \text{ min} \) if the next daily rest period will be a reduced one \((\mu_i^{dredrest} = 1)\) minus \( 60 + 45 = 105 \text{ minutes} \) if an extended daily driving time \((\mu_i^{extd3}((i,i+1)) = 1)\) was scheduled since the last daily rest period.

\[
(\alpha_{i,i+1}^{rest} = 1 \land \alpha_{i+1}^{rest} = 0) \Rightarrow E_{i+1}^t = E_{i+1}^{dd t} + 240 - 45 e_{i+1}^{bl} + 120 \mu_{i+1}^{dredrest} - 105 \mu_{i+1}^{extd3}
\]

In the case that a daily rest period is taken in vertex \( i + 1 \), the decision about the next daily rest period being a reduced one only influences \( L_{i+1} \) but not \( E_{i+1}^t \).

\[
(\alpha_{i,i+1}^{rest} = 1 \land \alpha_{i+1}^{rest} = 1) \Rightarrow E_{i+1}^t = E_{i+1}^{dd t} + 240 - 45 e_{i+1}^{bl} - 105 \mu_{i+1}^{extd3}
\]

If no daily rest period was taken on the arc \((i, i + 1)\), we obtain

\[
ee_{(i,i+1)}^{rest} = 0 \Rightarrow E_{i+1}^t = L_i - \Delta_{i,i+1}^{drive} - 45 A_{(i,i+1)}^{break} + 15 \mu_{(i,i+1)}^{upbreak}
\]

\(^{34}\) Recall that by constraints (2.5.41) a decision about a reduced rest period \( \mu_i^{dredrest} \) is linked with a daily rest period on the preceding arc or vertex.
We add the following linear constraints to our model:

\[
E_{i+1}^t \leq E_{i+1}^{ddt} + 240 - 45 \epsilon_{i+1}^{bt} + 120 t_{i+1}^{brest} - 105 \mu_{i,i+1}^{extd3} \\
\forall \ i = 0, \ldots, r - 2 \tag{2.5.134}
\]

\[
E_{i+1}^t \geq E_{i+1}^{ddt} + 240 - 45 \epsilon_{i+1}^{bt} + 120 t_{i+1}^{brest} - 105 \mu_{i,i+1}^{extd3} \\
- 900 \alpha_{i+1}^{rest} - 900 (1 - \alpha_{i,i+1}^{rest}) \\
\forall \ i = 0, \ldots, r - 2 \tag{2.5.135}
\]

\[
E_{i+1}^t \leq E_{i+1}^{ddt} + 240 - 45 \epsilon_{i+1}^{bt} - 105 \mu_{i,i+1}^{extd3} \\
+ 120 (1 - \alpha_{i+1}^{rest}) + 120 (1 - \alpha_{i,i+1}^{rest}) \\
\forall \ i = 0, \ldots, r - 2 \tag{2.5.136}
\]

\[
E_{i+1}^t \geq E_{i+1}^{ddt} + 240 - 45 \epsilon_{i+1}^{bt} - 105 \mu_{i,i+1}^{extd3} \\
- 780 (1 - \alpha_{i,i+1}^{rest}) \\
\forall \ i = 0, \ldots, r - 2 \tag{2.5.137}
\]

\[
E_{i+1}^t \leq L_i^t - \bar{\Delta}_{(i,i+1)}^{drive} - 45 A_{(i,i+1)}^{break} + 15 \mu_{(i,i+1)}^{upbreak} \\
+ (1200 + \frac{7}{6} \bar{\Delta}_{(i,i+1)}^{drive}) \alpha_{(i,i+1)}^{rest} \\
\forall \ i = 0, \ldots, r - 2 \tag{2.5.138}
\]

\[
E_{i+1}^t \geq L_i^t - \bar{\Delta}_{(i,i+1)}^{drive} - 45 A_{(i,i+1)}^{break} + 15 \mu_{(i,i+1)}^{upbreak} \\
\forall \ i = 0, \ldots, r - 2 \tag{2.5.139}
\]

Constraints (2.5.134) and (2.5.139) impose upper and lower bounds on \( E_{i+1}^t \), respectively. Additional lower and upper bounds that are used for determining appropriate big-M’s are:

- \( 0 \leq E_{i+1}^{ddt} \leq 540 \)
- \( 0 \leq L_i^t \leq 900 \)

For determining an appropriate value for \( M \) in (2.5.138), we first calculate an upper bound for the number of breaks on arc \((i, i+1)\). Therefore, recall (2.5.120) on page 69:

\[
E_{i+1}^{dt} \geq L_i^{dt} + 270 A_{(i,i+1)}^{break} - 270 l_i^{bn} + 270 A_{(i,i+1)}^{rest} - \bar{\Delta}_{(i,i+1)}^{drive} \\
- 210 \mu_{(i,i+1)}^{extd1} - 210 \mu_{(i,i+1)}^{extd2} - 210 \mu_{(i,i+1)}^{extd3} - 630 (1 - \lambda_i^{5})
\]

\[\Rightarrow \]

\[
270 A_{(i,i+1)}^{break} + 270 A_{(i,i+1)}^{rest} \leq E_{i+1}^{dt} + 270 l_i^{bn} + \bar{\Delta}_{(i,i+1)}^{drive} + 210 \mu_{(i,i+1)}^{extd1} + 210 \mu_{(i,i+1)}^{extd2} + 210 \mu_{(i,i+1)}^{extd3} + 630 (1 - \lambda_i^{5})
\]
2. Scheduling of driving times, breaks and rest periods

⇒

\[ 270 \: A_{(i,i+1)}^{\text{break}} \leq 2 \cdot 270 + \bar{\Delta}_{(i,i+1)}^{\text{drive}} + 3 \cdot 210 + 630 \]

⇔

\[ A_{(i,i+1)}^{\text{break}} \leq \frac{1800 + \bar{\Delta}_{(i,i+1)}^{\text{drive}}}{270} \]

We set \( M = 1200 + \frac{7}{6} \bar{\Delta}_{(i,i+1)}^{\text{drive}} \) and derive:

\[
\begin{align*}
M &= 1200 + \frac{7}{6} \bar{\Delta}_{(i,i+1)}^{\text{drive}} \\
\Leftrightarrow M &= 900 + \bar{\Delta}_{(i,i+1)}^{\text{drive}} + \frac{1800 + \bar{\Delta}_{(i,i+1)}^{\text{drive}}}{270} \\
\Rightarrow M &\geq E_{i+1}^{t} - L_{i}^{t} + \bar{\Delta}_{(i,i+1)}^{\text{drive}} + 45 A_{(i,i+1)}^{\text{break}} \\
\Leftrightarrow E_{i+1}^{t} &\leq L_{i}^{t} - \bar{\Delta}_{(i,i+1)}^{\text{drive}} - 45 A_{(i,i+1)}^{\text{break}} + M \\
\Rightarrow E_{i+1}^{t} &\leq L_{i}^{t} - \bar{\Delta}_{(i,i+1)}^{\text{drive}} - 45 A_{(i,i+1)}^{\text{break}} + 15 p_{(i,i+1)}^{\text{upbreak}} + M
\end{align*}
\]

So, \( M = 1200 + \frac{7}{6} \bar{\Delta}_{(i,i+1)}^{\text{drive}} \) is an appropriate choice, as constraints (2.5.138) become redundant in case daily rest periods are scheduled on the corresponding arcs \((i, i + 1)\).

2.5.18. Continuous driver status variables when leaving a vertex

The driver status when leaving vertex \( i \) results from the driver status when entering vertex \( i \) and the driver activities in vertex \( i \). Driver activities include resting (partially, full or reduced), taking a break (partially or full), waiting and loading/unloading goods. This interrelation is shown in Figure 2.4 on page 26.

In addition, in case a daily rest period is taken in vertex \( i \), the decision about the next daily rest period being a reduced one has to be taken into account. The two status variables \( L_{i}^{\text{dlt}} \) for the daily driving time left and \( L_{i}^{\text{dlt}} \) for the driving time left until the next break when leaving vertex \( i \) are bounded from above by \( L_{i}^{t} \), the maximum time allowed until the next daily rest period when leaving vertex \( i \). So we start with the discussion of the constraints concerning \( L_{i}^{t} \).

Driver activities in vertices are scheduled applying the rule that a resting activity is finished first before waiting and afterward loading and/or unloading goods may start.
2.5. Mathematical formulation

Time left until the next daily rest period

For determining \( L_i \), the time left until the next daily rest period when leaving vertex \( i \), we have to consider two cases:

- **Case 1:** A daily rest period is taken in vertex \( i \).
- **Case 2:** No daily period is taken in vertex \( i \).

In the first case, the time left until the next daily rest period equals 13 hours or 15 hours in case the next daily rest period is planned to be a reduced one, minus waiting time and minus the time needed for loading or unloading.

\[
\alpha_i^{\text{rest}} = 1 \Rightarrow L_i = 780 + 120 l_i^{\text{predrest}} - \Delta_i^{\text{wait}} - \bar{\Delta}_i^{\text{service}}
\]

In the second case we have to consider \( E_i \), the status when entering vertex \( i \neq 0 \), and additionally all other activities that are possible in a vertex. Note that if a first partial daily rest period is made, 3 + 9 = 12 hours have to be subtracted from the 24 hours time interval for the complete daily rest period to obtain the time for the other activities in this time interval. As one additional hour is needed for the daily rest period if it is split, 60 minutes are subtracted. The opportunity of substituting the last break on arc \((i - 1, i)\) by a first partial daily rest period \((\mu_i^{\text{prest}} = 1)\) is also taken into account. In that case, the first partial daily rest period migrates from vertex \( i \) to the arc \((i - 1, i)\) and it substitutes a break.

\[
\alpha_i^{\text{rest}} = 0 \Rightarrow L_i = E_i - \bar{\Delta}_i^{\text{service}} - 45 \alpha_i^{\text{break}} + 15 \mu_i^{\text{upbreak}} - 15 \alpha_i^{\text{break}} - 60 \alpha_i^{\text{prest}} + 45 \mu_i^{\text{prest}} - \Delta_i^{\text{wait}}
\]

In vertex 0, a (partial) break or a (first partial) daily rest period may take place or may be continued:

\[
\alpha_0^{\text{rest}} = 0 \Rightarrow L_0 = E_0 - (45 - \min(45, \overline{ubt} + 15 \overline{hpbl})) \alpha_0^{\text{break}} - (15 - \min(15, \overline{ubt})) \alpha_0^{\text{upbreak}} - (60 - \min(60, \overline{urt})) \alpha_0^{\text{prest}}
\]

Reformulating the above conditions by using the big-M approach and integrating upper and lower bounds we obtain the following linear constraints:

\[
L_i \leq 780 + 120 l_i^{\text{predrest}} - \Delta_i^{\text{wait}} - \bar{\Delta}_i^{\text{service}} \quad \forall \, i = 0, \ldots, r - 1 \quad (2.5.140)
\]

\[
L_i \geq 780 + 120 l_i^{\text{predrest}} - \Delta_i^{\text{wait}} - \bar{\Delta}_i^{\text{service}} - 900 (1 - \alpha_i^{\text{rest}}) \quad \forall \, i = 0, \ldots, r - 1 \quad (2.5.141)
\]

---

35 See rule 6 and optional rule 2 on page 19.
36 See also page 46.
2. Scheduling of driving times, breaks and rest periods

\[ L_i^t \leq E_i^{t} - \Delta_i^{\text{service}} - 45 \alpha_i^{\text{break}} + 15 \mu_i^{\text{break}} - 15 \alpha_i^{\text{prest}} - 60 \alpha_i^{\text{rest}} \]
\[ + 15 \mu_i^{\text{rest}} - \Delta_i^{\text{wait}} \]
\[ + 1020 \alpha_i^{\text{rest}} \]
\[ \forall i = 1, \ldots, r - 1 \]
\[ \text{(2.5.142)} \]

\[ L_i^t \geq E_i^{t} - \Delta_i^{\text{service}} - 45 \alpha_i^{\text{break}} + 15 \mu_i^{\text{break}} - 15 \alpha_i^{\text{prest}} - 60 \alpha_i^{\text{rest}} \]
\[ + 15 \mu_i^{\text{rest}} - \Delta_i^{\text{wait}} \]
\[ \forall i = 1, \ldots, r - 1 \]
\[ \text{(2.5.143)} \]

\[ L_0^t \leq E_0^{t} - (45 - \min(45, \bar{u}^{\text{bl}} + 15 \bar{h}^{\text{pb}})) \alpha_0^{\text{break}} - (15 - \min(15, \bar{u}^{\text{bl}})) \alpha_0^{\text{prest}} \]
\[ - (60 - \min(60, \bar{w}^{\text{rt}})) \alpha_0^{\text{rest}} \]
\[ + 1020 \alpha_0^{\text{rest}} \]
\[ \text{(2.5.144)} \]

\[ L_0^t \geq E_0^{t} - (45 - \min(45, \bar{u}^{\text{bl}} + 15 \bar{h}^{\text{pb}})) \alpha_0^{\text{break}} - (15 - \min(15, \bar{u}^{\text{bl}})) \alpha_0^{\text{prest}} \]
\[ - (60 - \min(60, \bar{w}^{\text{rt}})) \alpha_0^{\text{rest}} \]
\[ \text{(2.5.145)} \]

Constraints (2.5.140), (2.5.143) and (2.5.145) impose lower and upper bounds on \( L_i^t \). Additionally, \( L_1^t \leq 900 \) and \( E_i^{t} \leq 900 \) has to hold. Big-M’s for the other constraints were determined accordingly.

**Daily driving time left**

A set of auxiliary decision variables is needed to determine the driving time left until the next daily rest period, \( L_i^{\text{ddt}} \), that will now be introduced. We use the following bounds in conjunction with the big-M approach to set the corresponding constraints:

- \( 0 \leq L_i^t \leq 900 \)
- \( 0 \leq E_i^{\text{ddt}} \leq 540 \)
- \( 0 \leq E_i^{\text{ddt}} \leq 270 \)

The auxiliary decision variable \( \lambda_i^1 \) indicates if there is not enough time to take a break on arc \((i, i + 1)\) and completely use the daily driving time left (including a potential driving time extension), in case no daily rest period was made or to take a break and drive 540 min in total in case a daily rest period was taken. If \( \lambda_i^1 = 0 \) and no rest was taken, the time until the next daily rest period suffices to schedule an additional break on the next arc before the next daily rest period starts even if no such break is actually needed. Hence, the following conditions must hold:

\[(\alpha_i^{\text{rest}} = 0 \land L_i^t > E_i^{\text{ddt}} + 45 - 15 \mu_i^{\text{break}} + 60 \mu_i^{\text{extd}}) \Rightarrow \lambda_i^1 = 0 \]
\[(\alpha_i^{\text{rest}} = 0 \land L_i^t < E_i^{\text{ddt}} + 45 - 15 \mu_i^{\text{break}} + 60 \mu_i^{\text{extd}}) \Rightarrow \lambda_i^1 = 1 \]
\[(\alpha_i^{\text{rest}} = 1 \land L_i^t > 540 + 45) \Rightarrow \lambda_i^1 = 0 \]
\[(\alpha_i^{\text{rest}} = 1 \land L_i^t < 540 + 45) \Rightarrow \lambda_i^1 = 1 \]
This is induced by:

\[ 870 \left(1 - \lambda^2_i\right) \geq L_i^t - E_i^{dt} - 45 + 15 t_{\text{break}}^i - 60 \mu_i^{rest} - 870 \alpha_i^{rest} \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ -645 \lambda^3_i \leq L_i^t - E_i^{dt} - 45 + 15 t_{\text{break}}^i - 60 \mu_i^{rest} + 645 \alpha_i^{rest} \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ 315 \left(1 - \lambda^3_i\right) \geq L_i^t - 585 - 315 \left(1 - \alpha_i^{rest}\right) \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ -585 \lambda^3_i \leq L_i^t - 585 + 585 \left(1 - \alpha_i^{rest}\right) \]
\[ \forall i = 0, \ldots, r - 1 \]  

The auxiliary decision variable \( \lambda^2_i \) indicates if it is possible to take a break after \( E_i^{dt} \) or 270 minutes (in case a rest period, break or partial rest is scheduled in \( i \)) of driving after leaving vertex \( i \) before taking the next daily rest period.

\[ \left(\alpha_i^{rest} = 1 \land \alpha_i^{break} = 1 \lor (\alpha_i^{rest} = 1 \land \mu_i^{rest} = 0)\right) \land L_i^t > 315 \Rightarrow \lambda^2_i = 0 \]
\[ \left(\alpha_i^{rest} = 1 \land \alpha_i^{break} = 1 \lor (\alpha_i^{rest} = 1 \land \mu_i^{rest} = 0)\right) \land L_i^t < 315 \Rightarrow \lambda^2_i = 1 \]
\[ \left(\alpha_i^{rest} = 0 \land \alpha_i^{break} = 0 \lor (\alpha_i^{rest} = 0 \lor \mu_i^{rest} = 1)\right) \land L_i^t > E_i^{dt} + 45 - 15 t_{\text{break}}^i \Rightarrow \lambda_i^2 = 0 \]
\[ \left(\alpha_i^{rest} = 0 \land \alpha_i^{break} = 0 \lor (\alpha_i^{rest} = 0 \lor \mu_i^{rest} = 1)\right) \land L_i^t < E_i^{dt} + 45 - 15 t_{\text{break}}^i \Rightarrow \lambda_i^2 = 1 \]

We add the following constraints to our model:

\[ 585 \left(1 - \lambda^2_i\right) \geq L_i^t - 315 - 585 \left(1 - \alpha_i^{rest} - \alpha_i^{break} - \alpha_i^{rest} + \mu_i^{rest}\right) \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ -315 \lambda^2_i \leq L_i^t - 315 + 315 \left(1 - \alpha_i^{rest} - \alpha_i^{break} - \alpha_i^{rest} + \mu_i^{rest}\right) \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ 870 \left(1 - \lambda^3_i\right) \geq L_i^t - E_i^{dt} - 45 + 15 t_{\text{break}}^i - 870 \alpha_i^{rest} - 870 \alpha_i^{break} \]
\[ -870 \alpha_i^{rest} + 870 \mu_i^{prest} \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ -315 \lambda^2_i \leq L_i^t - E_i^{dt} - 45 + 15 t_{\text{break}}^i + 315 \alpha_i^{rest} + 315 \alpha_i^{break} \]
\[ + 315 \alpha_i^{rest} - 315 \mu_i^{prest} \]
\[ \forall i = 0, \ldots, r - 1 \]  

Note that only one resting activity per vertex is allowed if \( \mu_i^{prest} = 0 \). If \( \mu_i^{prest} = 1 \), a first partial break may still be taken (see constraints (2.5.99) and (2.5.100) on page 63).
2. Scheduling of driving times, breaks and rest periods

The auxiliary decision variable \( \lambda_3^i \) indicates if \( E_{dt}^i \) or 270 minutes (in case a resting activity was made in \( i \)) of driving until the next daily rest period are allowed due to \( L_t^i \). Hence,

\[
\begin{align*}
(\alpha_{i}^{\text{rest}} = 1 \lor \alpha_{i}^{\text{break}} = 1) \lor (\alpha_{i}^{\text{prest}} = 1 \land \mu_{i}^{\text{prest}} = 0) \land L_t^i > 270 & \Rightarrow \lambda_3^i = 0 \\
(\alpha_{i}^{\text{rest}} = 1 \lor \alpha_{i}^{\text{break}} = 1) \lor (\alpha_{i}^{\text{prest}} = 1 \land \mu_{i}^{\text{prest}} = 0) \land L_t^i < 270 & \Rightarrow \lambda_3^i = 1 \\
(\alpha_{i}^{\text{rest}} = 0 \land \alpha_{i}^{\text{break}} = 0) \lor (\alpha_{i}^{\text{prest}} = 0 \lor \mu_{i}^{\text{prest}} = 1) \land L_t^i > E_{dt}^i & \Rightarrow \lambda_3^i = 0 \\
(\alpha_{i}^{\text{rest}} = 0 \land \alpha_{i}^{\text{break}} = 0) \lor (\alpha_{i}^{\text{prest}} = 0 \lor \mu_{i}^{\text{prest}} = 1) \land L_t^i < E_{dt}^i & \Rightarrow \lambda_3^i = 1
\end{align*}
\]

The above conditions are enforced by

\[
\begin{align*}
630 (1 - \lambda_3^i) & \geq L_t^i - 270 - 630 (1 - \alpha_{i}^{\text{rest}} - \alpha_{i}^{\text{break}} - \alpha_{i}^{\text{prest}} + \mu_{i}^{\text{prest}}) \\
\forall i = 0, \ldots, r - 1 & \quad (2.5.154) \\
-270 \lambda_3^i & \leq L_t^i - 270 + 270 (1 - \alpha_{i}^{\text{rest}} - \alpha_{i}^{\text{break}} - \alpha_{i}^{\text{prest}} + \mu_{i}^{\text{prest}}) \\
\forall i = 0, \ldots, r - 1 & \quad (2.5.155) \\
900 (1 - \lambda_3^i) & \geq L_t^i - E_{dt}^i - 900 \alpha_{i}^{\text{rest}} - 900 \alpha_{i}^{\text{break}} - 900 \alpha_{i}^{\text{prest}} + 900 \mu_{i}^{\text{prest}} \\
\forall i = 0, \ldots, r - 1 & \quad (2.5.156) \\
-270 \lambda_3^i & \leq L_t^i - E_{dt}^i + 270 \alpha_{i}^{\text{rest}} + 270 \alpha_{i}^{\text{break}} + 270 \alpha_{i}^{\text{prest}} - 270 \mu_{i}^{\text{prest}} \\
\forall i = 0, \ldots, r - 1 & \quad (2.5.157)
\end{align*}
\]

Since

\[
\lambda_3^i = 1 \Rightarrow \lambda_2^i = 1
\]

must hold, we add the constraints

\[
\lambda_2^i \geq \lambda_3^i \quad \forall i = 0, \ldots, r - 1 \quad (2.5.158)
\]

The last auxiliary decision variable \( \lambda_6^i \) is needed to check if in case that no daily rest period is taken in \( i \), the time left until the next daily rest period when leaving vertex \( i \), \( L_t^i \), is larger than the daily driving time left when entering vertex \( i \) plus 60 minutes if a driving time extension is used (\( \mu_{i}^{\text{extd}} = 1 \)). If a daily rest period is taken in \( i \), \( \lambda_6^i \) is set to be equal to 1.

\[
\begin{align*}
(L_t^i > E_{dt}^i + 60 \mu_{i}^{\text{extd}} \land \alpha_{i}^{\text{rest}} = 0) & \Rightarrow \lambda_6^i = 0 \\
(L_t^i < E_{dt}^i + 60 \mu_{i}^{\text{extd}} \land \alpha_{i}^{\text{rest}} = 0) & \Rightarrow \lambda_6^i = 1 \\
\alpha_{i}^{\text{rest}} = 1 & \Rightarrow \lambda_6^i = 1
\end{align*}
\]
We add the constraints:

\[
900 \ (1 - \lambda_i^6) \geq L_i^t - E_i^{ddt} - 60 \ \mu_i^{extd} - 900 \ \alpha_i^{rest} \quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.159)
\]

\[
-540 \ \lambda_i^6 \leq L_i^t - E_i^{ddt} - 60 \ \mu_i^{extd} + 540 \ \alpha_i^{rest} \quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.160)
\]

\[
\lambda_i^6 \geq \alpha_i^{rest} \quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.161)
\]

Now, we can determine the driving time left until the next daily rest period, \( L_i^{ddt} \), when leaving vertex \( i \). First of all we consider the case that \( \lambda_i^1 = 0 \) and a daily rest period is taken in \( i \). The following condition must be satisfied:

\[
(\lambda_i^1 = 0 \land \alpha_i^{rest} = 1) \Rightarrow L_i^{ddt} = 540
\]

As \( \lambda_i^1 = 0 \), the time until the next daily rest period suffices for the maximum daily driving time of 9 hours and a 45 minute break that has to be taken after at most 4.5 hours of uninterrupted driving. For the case that no daily rest period is taken in \( i \) and \( \lambda_i^1 = 0 \), we have to impose that

\[
(\lambda_i^1 = 0 \land \alpha_i^{rest} = 0) \Rightarrow L_i^{ddt} = E_i^{ddt} + 60 \ \mu_i^{extd}
\]

Here, \( \lambda_i^1 \) indicates that there is enough time until the next daily rest period left to completely use the daily driving time left when entering \( i \), including a 45 minute break (no matter if needed or not) plus 60 minutes if a driving time extension is used.

If \( \lambda_i^1 = 1 \) and \( \lambda_i^2 = 0 \), there is enough time to schedule 270 minutes of driving and to take a break (second part or full) right after leaving vertex \( i \) if a resting activity that extends the driving time until the next break or daily rest period takes place in \( i \). If no such resting activity takes place, there is enough time to schedule \( E_i^{ddt} \) minutes of driving and a break (second part or full). The time until the next rest period, \( L_i^t \), does not exceed the time needed to completely use 9 hours driving time gained by taking a daily rest period in \( i \) or to use the daily driving time left when entering vertex \( i \), respectively. In that case, the daily driving time left \( L_i^{ddt} \) when leaving \( i \) equals the time left until the next daily rest period \( L_i^t \) when leaving \( i \) minus the time needed for a break (full or second part).

\[
(\lambda_i^1 = 1 \land \lambda_i^2 = 0) \Rightarrow L_i^{ddt} = L_i^t - 45 + 15 \ l_i^{break}
\]

If \( \lambda_i^2 = 1 \) and \( \lambda_i^3 = 0 \), we differentiate between the case with a resting activity that extends the driving time in vertex \( i \) and the case without. We will first take a look at the case with no resting activity in \( i \). Because of \( \lambda_i^2 = 1 \), the time until the next daily rest period does not suffice to schedule \( E_i^{ddt} \) minutes of driving until the next daily rest period and to take a break. But as \( \lambda_i^3 = 0 \), it follows that \( E_i^{dt} \leq L_i^t \leq E_i^{dt} + 45 - 15 \ l_i^{break} \). That means, \( E_i^{dt} \) minutes of driving are possible but a break to extend the driving time is not possible, as it would not fit in the time interval between the last and the subsequent daily rest period.

\[
(\lambda_i^2 = 1 \land \lambda_i^3 = 0 \land \alpha_i^{rest} = 0 \land \alpha_i^{break} = 0 \land (\alpha_i^{prest} = 0 \lor \mu_i^{prest} = 1) \Rightarrow L_i^{ddt} = E_i^{dt}
\]
If a resting activity that extends the driving time is made in vertex \( i \), we differentiate between the case that \( \lambda_i^6 = 1 \) and \( \lambda_i^6 = 0 \). If \( \lambda_i^6 = 1 \), 270 minutes of driving until the next daily rest period are possible, otherwise, \( E_{i}^{ddt} \) minutes are possible plus 60 minutes if the daily driving time is extended with a corresponding break in vertex \( i \).

\[
(\lambda_i^2 = 1 \land \lambda_i^3 = 0 \land (\alpha_i^{rest} = 1 \lor \alpha_i^{break} = 1 \lor (\alpha_i^{prest} = 1 \land \mu_i^{prest} = 0)) \land \lambda_i^6 = 1) \\
\Rightarrow L_i^{ddt} = 270 \\
(\lambda_i^2 = 1 \land \lambda_i^3 = 0 \land (\alpha_i^{rest} = 1 \lor \alpha_i^{break} = 1 \lor (\alpha_i^{prest} = 1 \land \mu_i^{prest} = 0)) \land \lambda_i^6 = 0) \\
\Rightarrow L_i^{ddt} = E_{i}^{ddt} + 60 \mu_i^{extd}
\]

In case \( \lambda_i^3 = 1 \), the time left until the next daily rest period does not suffice to drive 270 minutes, in case a resting activity that extends the driving time was made in \( i \). Moreover, it does not suffice to drive \( E_{i}^{ddt} \) minutes in case no resting activity was made. Depending on \( \lambda_i^6 \), we can determine \( L_i^{ddt} \) as follows:

\[
(\lambda_i^3 = 1 \land \lambda_i^6 = 1) \Rightarrow L_i^{ddt} = L_i^t \\
(\lambda_i^3 = 1 \land \lambda_i^6 = 0) \Rightarrow L_i^{ddt} = E_{i}^{ddt} + 60 \mu_i^{extd}
\]

Again, we use the big-M approach and integrate upper and lower bounds to derive linear constraints. Redundant constraints like \( L_i^{ddt} \leq 540 \) are omitted.

\[
L_i^{ddt} \geq 540 - 540 \lambda_i^1 - 540 (1 - \alpha_i^{rest}) \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.162) \\
L_i^{ddt} \leq E_{i}^{ddt} + 60 \mu_i^{extd} + 540 \alpha_i^{rest} \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.163) \\
L_i^{ddt} \geq E_{i}^{ddt} + 60 \mu_i^{extd} - 600 \alpha_i^{rest} - 600 \lambda_i^1 \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.164) \\
L_i^{ddt} \leq L_i^t - 45 + 15 t_i^{break} + 45 (1 - \lambda_i^1) + 45 \lambda_i^2 \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.165) \\
L_i^{ddt} \geq L_i^t - 45 + 15 t_i^{break} - 855 (1 - \lambda_i^1) - 855 \lambda_i^2 \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.166) \\
L_i^{ddt} \leq E_{i}^{ddt} + 540 (1 - \lambda_i^2) + 540 \lambda_i^3 + 540 \alpha_i^{break} + 540 \alpha_i^{rest} + 540 \alpha_i^{prest} \\
\quad - 540 \mu_i^{prest} \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.167) \\
L_i^{ddt} \geq E_{i}^{ddt} - 270 (1 - \lambda_i^2) - 270 \lambda_i^3 - 270 \alpha_i^{break} - 270 \alpha_i^{rest} - 270 \alpha_i^{prest} \\
\quad + 270 \mu_i^{prest} \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.168) \\
L_i^{ddt} \leq 270 + 270 (1 - \lambda_i^2) \\
\quad \forall \ i = 0, \ldots, r - 1 \quad (2.5.169)
\]
2.5. Mathematical formulation

\[ L_{i}^{ddt} \geq 270 - 270 (1 - \lambda_i^2) - 270 \lambda_i^3 - 270 (1 - \alpha_i^{break} - \alpha_i^{rest} - \alpha_i^{prest} + \mu_i^{prest}) \]
\[ \quad - 270 (1 - \lambda_i^6) \quad \forall i = 0, \ldots, r - 1 \]  
\[ (2.5.170) \]

\[ L_{i}^{ddt} \leq E_{i}^{ddt} + 60 \mu_i^{extd} + 540 (1 - \lambda_i^2) + 540 \lambda_i^3 + 540 (1 - \alpha_i^{break} - \alpha_i^{prest} + \mu_i^{prest}) + 540 \lambda_i^6 \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ (2.5.171) \]

\[ L_{i}^{ddt} \geq E_{i}^{ddt} + 60 \mu_i^{extd} - 540 (1 - \lambda_i^2) - 540 \lambda_i^3 \]
\[ - 540 (1 - \alpha_i^{break} - \alpha_i^{prest} + \mu_i^{prest}) - 540 \lambda_i^6 \]
\[ \forall i = 0, \ldots, r - 1 \]  
\[ (2.5.172) \]

Driving time left until the next break or daily rest period

Using \( L_{i}^{ddt} \) as an upper bound, the driving time left until the next break or daily rest period \( L_{i}^{dt} \) when leaving vertex \( i \) can now be easily determined.

If a resting activity was made in vertex \( i \), \( L_{i}^{dt} \) depends on whether \( L_{i}^{ddt} > 270 \) or not. Therefore, we introduce the auxiliary variable \( \lambda_i^4 \):

\[ L_{i}^{ddt} > 270 \Rightarrow \lambda_i^4 = 1 \]
\[ L_{i}^{ddt} < 270 \Rightarrow \lambda_i^4 = 0 \]

We formulate this in our model as follows:

\[ 270 \lambda_i^4 \geq L_{i}^{ddt} - 270 \quad \forall i = 0, \ldots, r - 1 \]  
\[ (2.5.176) \]

\[ 270 (\lambda_i^4 - 1) \leq L_{i}^{ddt} - 270 \quad \forall i = 0, \ldots, r - 1 \]  
\[ (2.5.177) \]

With \( \lambda_i^4 \) we obtain the following conditions:

\[ (\alpha_i^{rest} = 1 \lor \alpha_i^{break} = 1 \lor (\alpha_i^{prest} = 1 \land \mu_i^{prest} = 0) \land \lambda_i^4 = 1) \Rightarrow L_{i}^{dt} = 270 \]
\[ (\alpha_i^{rest} = 1 \lor \alpha_i^{break} = 1 \lor (\alpha_i^{prest} = 1 \land \mu_i^{prest} = 0) \land \lambda_i^4 = 0) \Rightarrow L_{i}^{dt} = L_{i}^{ddt} \]
In case no extending rest activity is made in vertex $i$, $E_i^{dt}$ depends on whether $L_i^t > E_i^{dt}$ or not. In the last section, we introduced the variable $\lambda_i^3$ that is equal to zero if $L_i^t > E_i^{dt}$, and equal to one if $L_i^t < E_i^{dt}$ in case no extending rest activity is made. We obtain the following conditions:

$$(\alpha_i^{\text{rest}} = 0 \land \alpha_i^{\text{break}} = 0 \land (\alpha_i^{\text{prest}} = 0 \lor \mu_i^{\text{prest}} = 1) \land \lambda_i^3 = 1) \Rightarrow L_i^{dt} = L_i^{ddt}$$

$$(\alpha_i^{\text{rest}} = 0 \land \alpha_i^{\text{break}} = 0 \land (\alpha_i^{\text{prest}} = 0 \lor \mu_i^{\text{prest}} = 1) \land \lambda_i^3 = 0) \Rightarrow L_i^{dt} = E_i^{dt}$$

These logical conditions are induced by the following constraints:

$$L_i^{dt} \geq 270 - 270 (1 - \lambda_i^4) - 270 (1 - \alpha_i^{\text{rest}} - \alpha_i^{\text{break}} - \alpha_i^{\text{prest}} + \mu_i^{\text{prest}}) \quad \forall i = 0, \ldots, r - 1$$

(2.5.178)

$$L_i^{dt} \leq L_i^{ddt} \quad \forall i = 0, \ldots, r - 1$$

(2.5.179)

$$L_i^{dt} \geq L_i^{dt} - 540 \lambda_i^4 - 540 (1 - \alpha_i^{\text{rest}} - \alpha_i^{\text{break}} - \alpha_i^{\text{prest}} + \mu_i^{\text{prest}}) \quad \forall i = 0, \ldots, r - 1$$

(2.5.180)

$$L_i^{dt} \geq L_i^{dt} - 540 (1 - \lambda_i^3) - 540 \alpha_i^{\text{rest}} - 540 \alpha_i^{\text{break}} - 540 \alpha_i^{\text{prest}} + 540 \mu_i^{\text{prest}} \quad \forall i = 0, \ldots, r - 1$$

(2.5.181)

$$L_i^{dt} \leq E_i^{dt} + 270 \alpha_i^{\text{rest}} + 270 \alpha_i^{\text{break}} + 270 \alpha_i^{\text{prest}} - 270 \mu_i^{\text{prest}} \quad \forall i = 0, \ldots, r - 1$$

(2.5.182)

$$L_i^{dt} \geq E_i^{dt} - 270 \lambda_i^3 \quad \forall i = 0, \ldots, r - 1$$

(2.5.183)
2.5. Mathematical formulation

2.5.19. Objective functions

The main objective is to minimize the total lateness. The completion time, i.e. the overall schedule duration until the last customer is serviced and the last stop is reached is important, too. Other criteria are relevant for the quality of a solution in practice, but are considered less important. We chose a combination of strategies for this multicriteria optimization problem. At first, two objective functions were created. One for lateness and completion time and one for all other criteria. Within both objective functions different weights were provided for each of the objectives. As the criteria in the first objective function were considered to be more important than those in the second one, a lexicographic ordering was applied. The two objective functions will be presented in the following.

Objective function 1

Penalized lateness for violating time windows is minimized along with the completion time.

\[
\text{Minimize } \text{start}_{r-1} + \sum_{i=1}^{r-1} P \cdot \Delta_{i}^{\text{late}}
\]

with \( P \) as a user-specified penalty constant. Note that the begin of service time \( \text{start}_{r-1} \) is also penalized (with factor 1) in the objective function. As we consider lateness to be the most important optimization criterion, the penalty of 1 minute lateness is more important (i.e. has higher weight) than the penalty of the latest completion time possible. The maximum time between two weekly rest periods is \( 6 \cdot 24 \) h = 8640 min. The schedule starts directly after finishing a weekly rest period (time 0). The latest completion time \( \text{start}_{r-1} + \Delta_{r-1}^{\text{service}} \) is therefore less than 8640. We set \( P = 8640 \) neglecting the time needed for loading and/or unloading goods at the last customer location, \( \Delta_{r-1}^{\text{service}} \).

Objective function 2

It is important that dispatchers and drivers accept the schedules generated as otherwise they will not adopt them. Different criteria that are important for the quality of a solution and thus the compliance with the resulting schedule have not been taken into account until now. As dispatchers, drivers and anyone else that tries to analyze the schedule would be confused if the schedule contained, for example, unnecessary early daily rest periods or if waiting time would occur even though the lower boundary of the chosen time window had already been exceeded, such cases have to be eliminated to obtain more comprehensible solutions. Furthermore, it may be advantageous if the solution does not exploit driving time extensions or reduced daily rest periods completely. This is, for example, the case if the planning horizon does not comprise the whole week. Moreover, if unexpected events during the execution of the plan such as traffic congestion or delays in loading or unloading the
goods occur, using the optional rules may help to compensate the additional time needed. So the idea is to only use optional rules if there is any benefit. In other words, not making use of the optional rules would worsen lateness and/or completion time.

The objective function (2.5.185) contains different criteria needed to meet the objectives described above. They are provided with different weights that may be customized.

\[
\text{Minimize } \sum_{i=1}^{r-1} \sum_{z=0}^{\text{notTW}_i-1} 10 (z + r - i) \ tw_{iz} + \sum_{i=0}^{r-1} \text{start}_i \\
+ \sum_{i=0}^{r-2} 10 (r - i) (\mu_{\text{earlydr}1}^{(i,i+1)} + \mu_{\text{earlydr}2}^{(i,i+1)}) \\
+ \sum_{i=0}^{r-1} 10 (r - i) (\alpha_{i}^{\text{break}} + \alpha_{i}^{\text{rest}}) \\
+ \sum_{i=0}^{r-1} 20 \Delta_{i}^{\text{wait}} \\
+ \sum_{i=0}^{r-2} 30 (r - i) \mu_{i}^{\text{redrest}} + \sum_{i=0}^{r-1} 40 (r - i) \mu_{i}^{\text{redrest}} \\
+ \sum_{i=0}^{r-2} 50 (r - i) \mu_{i}^{\text{extd}2} + 60 (r - i) \mu_{i}^{\text{extd}1} + 60 (r - i) \mu_{i}^{\text{extd}3} \\
+ \sum_{i=0}^{r-1} 60 (r - i) \mu_{i}^{\text{extd}} \\
(2.5.185)
\]

The first line of the objective function deals with the choice of time windows and the start of loading and/or unloading the vehicle. Note that activities in the near future are more certain while activities in the far future may have to be re-planned more often as the original planning conditions may change substantially. Thus, it is desirable, especially at the beginning of the planning horizon, that the time windows chosen are rather early than late to avoid wasted time at the beginning of the schedule.\(^37\) In case of unexpected events, this generally leaves more possibilities for re-planning steps. The start of loading and/or unloading the vehicle is also preferred to start as early as possible for the same reasons.

The second line of the objective function addresses early daily rest periods, i.e. daily rest periods that take place although only 4.5 hours of driving took place since the previous daily rest period.\(^38\) In case of long-haul freight transportation loading and/or unloading the vehicle constitute only a small proportion of the overall time between two daily rest periods. This means that an early daily rest period often leads to a very short time interval.

\(^{37}\) Time windows have to be sorted by their start time and shall not overlap.

\(^{38}\) The cases where this can be advantageous are described in Section 2.5.14.
between two daily rest periods which primarily contains driving activities. Hence, planning an early daily rest period often means for the driver having to accept an irregular time interval for recuperation (including sleeping) which is not desirable as this interferes with his biorhythm. This is the reason why early daily rest periods are penalized.

Partial breaks and partial daily rest periods are considered in the third line. As far as possible, they should only be scheduled if there is any benefit, i.e. an improvement considering the first objective function value. This rule leads to more comprehensible schedules as the driver only sees partial breaks and rest periods that are absolutely necessary to be on time and not to increase the schedule duration. it is not intended to keep drivers from using planned waiting times for additional breaks, which can be advantageous in case of unexpected changes.

Waiting time is penalized in line four. This avoids that waiting time reaches into the chosen time window unnecessarily and has the positive side effect that daily rest periods are extended to compensate for waiting time.

Lines five to seven deal with optional rules that may only be applied for a maximum number of times each week or between two weekly rest periods. Thus, weights are chosen to be greater than for the other considered criteria. Again, the goal is to obtain more freedom the closer the schedule gets to its end, leaving room for changes due to unforeseen events and to schedule additional requests.

Reduced daily rest periods can take place at most three times between two weekly rest periods whereas extended daily driving times are allowed only twice a week. Thus, the decision was made to penalize extended daily driving times stronger than reduced daily rest periods. Additionally, it is preferred to take a daily rest period between stop \( i \) and \( i + 1 \) instead taking a daily rest period at arrival at stop \( i \).

Considering driving time extensions, weights are chosen to prefer those that take place between two daily rest periods and between two stops, as in these cases, the maximum of one hour driving time extension is used up.\(^{39}\) For the other types of driving time extensions considered in the MILP model, extensions may be restricted to less than one hour due to the maximum time between two daily rest periods. In addition to driving activities, waiting times, times needed for resting activities, and times needed for loading and/or unloading have to be taken into account.

### 2.5.20. The lexicographic approach

The constraints described in the previous sections together with the two objective functions presented in Section 2.5.19 form the MILP model with the consideration of optional rules. Two MILP submodels are created to be able to solve the problem sequentially. The objective function (2.5.184) together with the constraints described form one submodel. This submodel is solved in the first optimization step (using a MILP solver). In the second optimization step, the objective function of the previous step is transformed into constraint

\(^{39}\) For the different types of driving time extensions see page 51.
2. Scheduling of driving times, breaks and rest periods

(2.5.186) with the previous objective function value \( z^* \) as an upper bound such that the weighted sum of overall lateness and completion time is prevented from increasing.

\[
start_{r-1} + \sum_{i=1}^{r-1} P \cdot \Delta_{late}^i \leq z^* \tag{2.5.186}
\]

Together with the objective function (2.5.185) and the constraints from the previous optimization step we obtain the second submodel which is then solved (with a MILP solver). The solution obtained is the solution of the complete MILP model.

2.5.21. The MILP model without optional rules

If no optional rules should be taken into account, the MILP model can easily be adapted. By adding the following constraints, all optional rules are switched off.

\[
\begin{align*}
\mu_{i,i+1}^{\text{rest}} &= 0 & \forall i &= 0,\ldots,r-2 \\
\mu_{i,i+1}^{\text{extd1}} &= 0 & \forall i &= 0,\ldots,r-2 \\
\mu_{i,i+1}^{\text{extd2}} &= 0 & \forall i &= 0,\ldots,r-2 \\
\mu_{i,i+1}^{\text{extd3}} &= 0 & \forall i &= 0,\ldots,r-2 \\
\mu_{i,i+1}^{\text{upbreak}} &= 0 & \forall i &= 0,\ldots,r-2 \\
\mu_{i,i+1}^{\text{prest}} &= 0 & \forall i &= 0,\ldots,r-2 \\
\mu_i^{\text{rest}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\alpha_i^{\text{prest}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\alpha_i^{\text{upbreak}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{extd}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{dredrest}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{upbreak}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{break}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{prest}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{dredrest}} &= 0 & \forall i &= 0,\ldots,r-1 \\
\mu_i^{\text{extd}} &= 0 & \forall i &= 0,\ldots,r-1 
\end{align*}
\]

These constraints are needed later when the influence of the optional rules on the run time and the objective function value is studied. Additionally, we will later use the solution of the MILP model with disabled optional rules as an upper bound for the
optimal objective function value of the model with optional rules (only considering objective function (2.5.184)).

2.6. Transformation into a driver schedule

The solution obtained for the MILP model contains values for many variables that have to be considered in combination to be able to derive driver activities that can be executed in practice. To show that this is a non-trivial task, in Figure 2.10 we consider exemplary solution values representing the driver status and activities between two consecutive customer stops.

![Diagram showing driver activities and time slots]

Figure 2.10.: Transformation is necessary

Note that in the different steps, we only consider a part of the solution. Assume that a driving time of 6 hours (360 minutes) to get from stop $i$ to $i+1$, $\Delta_{(i,i+1)}^{drive}$, is given. Consider the values determined for

- the driving time left until the next break or daily rest period when leaving $i$ ($L_i = 30$),
- the driving time left until the next daily rest period when leaving $i$ ($L_i = 120$),
the number of breaks between stop $i$ and $i+1$ ($A_{i,i+1} = 1$) and

- the number of daily rest periods between stop $i$ and $i+1$ ($A_{i,i+1} = 1$).

Given these values, a possible extract of the driver schedule with activities between stops $i$ and $i+1$ may look like the first schedule depicted in Figure 2.10: the driving time left until the next break or daily rest period is exhausted and then a break is taken. Afterward, the driver uses up the remaining 90 minutes (120 minutes - 30 minutes) of the daily driving time continuing with a daily rest period with a duration of 11 hours. Interruption-free, 4 hours of driving follow before reaching stop $i+1$.

Now assume, that we overlooked an early daily rest period between stop $i$ and $i+1$ ($\mu_{\text{early dr}} = 1$) that is given by the MILP model solution. This additional information implicates, that the first resting activity between stop $i$ and $i+1$ is a daily rest period, as shown in the second schedule. This means that the first attempt to manually transform the model solution has lead us to a wrong partial driver schedule.

If we integrate the variable value of the cumulative duration of daily rest periods between stops $i$ and $i+1$, $\Delta_{i,i+1}$, which is 9 hours and 10 minutes (550 minutes) in our example, together with the information that the daily rest period taken is a reduced one (the number of reduced daily rest periods on arc $(i, i+1)$ is equal to 1, $\mu_{\text{redrest}} = 1$), we recognize that the additional information invalidates the schedule that was chosen in the previous step. The third schedule in Figure 2.10 remains a possible option.

Adding with $\mu_{i+1} = 1$ the information that the last break between stops $i$ and $i+1$ is substituted by a first partial daily rest period, again the number of possible schedules is reduced, leaving the last schedule depicted in Figure 2.10 as a possible outcome.

The example shows that a manual transformation has to take into account many variable values and is thus very difficult and error prone. Thus, a transformation algorithm (Algorithm 1, pages 91 to 100) is presented that transforms the MILP model solution into a detailed driver schedule.

In the algorithm, the method `addActivity` ($(i, i+1), \langle \text{type} \rangle, \langle \text{duration} \rangle$) is called to add an activity of type $\langle \text{type} \rangle$ and duration $\langle \text{duration} \rangle$ as last activity on arc $(i, i+1)$ that in this case represents the time interval between the start after unloading/loading goods at customer stop $i$ and the end of loading/unloading goods at customer stop $i+1$. Activity types are represented by the constants "drive", "rest" and "wait".

The algorithm starts with the determination of driver activities at the origin (lines 8 to 36). Therefore, the variable values for decisions on a daily rest period ($\alpha_{0}^{\text{rest}}$), a break ($\alpha_{0}^{\text{break}}$), a first partial break ($\alpha_{0}^{\text{pbreak}}$), a first partial daily rest period ($\alpha_{0}^{\text{pbreak}}$), and waiting time ($\Delta_{0}^{\text{wait}}$) are taken into account. When a resting activity has already started at the beginning of the planning horizon ($\overline{urt} > 0$ or $\overline{ubt} > 0$), the time spent for this activity is considered if the activity is continued. A first partial break already taken before the
beginning of the planning horizon ($h_{pb} = 1$) influences the remaining duration of a break at the origin.

The algorithm continues determining the activities for each pair of consecutive stops $(i, i+1)$ (lines 38 to 304). In each iteration, there are 6 major steps, which are described in the following.

In the first step (lines 43 to 115), activities either before the first daily rest period on the arc $(i, i + 1)$ or until the next stop is reached are planned, depending on what happens earlier. Therefore, the driving time left until the next break or daily rest period when leaving stop $i$, $L_i$, the driving time left until the next daily rest period when leaving stop $i$, $L_{ddt}^i$, and the time needed to traverse the arc $(i, i + 1)$ are taken into account. Early daily rest periods, partial breaks, partial daily rest periods and extended daily driving times are considered in this step as well. Note that $helpPartialRest$ indicates if a partial daily rest period between stops $i$ and $i + 1$ has already been integrated into the schedule.

In the second step (lines 117 to 138), if there is at least one daily rest period on arc $(i, i + 1)$ ($\star_{rest} A_{(i, i+1)}^{i+1} \geq 1$), the first daily rest period is scheduled, its duration depending on the overall duration of daily rest periods on this arc and the number of daily rest periods to take place. A partial daily rest period still to exhaust when leaving stop $i$ ($\star_{l prest} l_{rest}^i = 1$) or if the decision is made that the first daily rest period between stops $i$ and $i + 1$ should be a reduced one ($\star_{l drest} l_{drest}^i = 1$), also influences the duration. Note that in case the driving time until the next daily rest period when arriving at stop $i + 1$ is equal to 540 ($\star_{E ddt} E_{ddt}^{i+1} = 540$), the last daily rest period between stops $i$ and $i + 1$ is postponed to the vertex $i + 1$. If an additional daily rest period then takes place at stop $i + 1$, the durations of the postponed daily rest period ($\star_{restCarryover2}$, determined in lines 252 to 258) and of the daily rest period already planned for stop $i + 1$ are summed up. This prevents that two consecutive daily rest periods without another activity in between are scheduled in favor for one long daily rest period which is identifiable as such.

If unforeseen events force plan changes, overlong daily rest periods already taken may limit the set of possibilities when re-planning. Hence, it is advantageous to take longer daily rest periods as late as possible.\(^{41}\) Thus, in step two, the variable $\star_{restCarryover1}$ is determined as difference between the cumulative duration of all daily rest periods $\Delta_{(i, i+1)}^{rest}$ and the cumulative duration if daily rest periods are all considered to have either a regular duration of 11 hours or 9 hours in case of second partial or reduced daily rest periods. This difference is added to the duration of the last daily rest period before loading and/or unloading at stop $i + 1$ (see lines 132 to 134, 195 to 197, and 252 to 264).

\(^{40}\) Note that the maximum driving time until the next daily rest period when arriving at stop $i + 1$ (without taking into account optional rules, which are expressed by additional variables) is less than or equal to nine hours ($\star_{E ddt} E_{ddt}^{i+1} \leq 540$). If the driving duration to traverse the arc $(i, i + 1)$ is greater than 0, $E_{ddt}^{i+1} = 540$ means that the last activity planned between stops $i$ and $i + 1$ is a daily rest period.

\(^{41}\) On the durations of daily rest periods see Section 2.5.9.
In step three (lines 140 to 182), the driver activities between the first and the last daily rest period are planned. Information on the number of daily rest periods planned, if an early daily rest period is planned as last resting activity, reduced daily rest periods and extended daily driving times are taken into account. Note that extended daily driving times between the first and the last daily rest period are always scheduled as late as possible. The same applies to reduced daily rest periods.

In case that more than one daily rest period is taken between stops \( i \) and \( i + 1 \), the last daily rest period is scheduled in step four (lines 184 to 201), taking into account if a reduced daily rest period is still left to be scheduled for the arc \((i, i + 1)\). Again, similar to step 2, the possibility that one daily rest period more should be taken on arc \((i, i + 1)\) than actually necessary to traverse the arc (indicated by \( E_{i+1} = 540 \)) in combination with a daily rest period at stop \( i + 1 \) is excluded.

In step five (lines 202 to 244), the driver activities after the last daily rest period between stops \( i \) and \( i + 1 \) are planned if at least one daily rest period is scheduled on the arc \((i, i + 1)\) (again excluding the case \( E_{i+1} = 540 \)). Therefore, the driving time left until the next daily rest period when arriving at stop \( i + 1 \), \( E_{i+1} \) is considered. If \( E_{i+1} \) is less than 270 (4:30 h), more than 4:30 hours of driving must have taken place since the last daily rest period when arriving at stop \( i + 1 \). A break or a partial daily rest period is planned accordingly, depending on if the possibility is chosen to substitute the last break on the arc \((i, i + 1)\) by a first partial daily rest period and/or a daily driving time extension after the last daily rest period is planned. This is taken into account by considering the combination of the variable values of \( \alpha_{i+1}^{\text{prest}} \), \( \mu_{i+1}^{\text{extd}} \) and \( \mu_{(i,i+1)}^{\text{extd}} \), where \( \alpha_{i+1}^{\text{prest}} \) indicates if a partial daily rest period is planned to take place at stop \( i + 1 \) (which may be shifted to the arc \((i, i + 1)\) depending on the other variables mentioned), \( \mu_{i+1}^{\text{extd}} \) indicates if a driving time extension is initiated by a break at stop \( i + 1 \) and \( \mu_{(i,i+1)}^{\text{extd}} \) is equal to one, if a driving time extension starts between the last daily rest period on arc \((i, i + 1)\) and the next stop \( i + 1 \).

The last step (lines 246 to 302) deals with scheduling driver activities after the arrival at stop \( i + 1 \). Possible activities are a daily rest period (\( \alpha_{i+1}^{\text{rest}} = 1 \) and/or \( \text{restCarryover}_2 > 0 \)), taking a break (\( \alpha_{i+1}^{\text{break}} = 1 \)), waiting (duration: \( \Delta_{i+1} \)) and loading and/or unloading the vehicle (duration: \( \text{start}_{i+1} \)). The first or second part of a break (\( \alpha_{i+1}^{\text{break}} = 1 \) or \( \alpha_{i+1}^{\text{break}} = 1 \land \lnot l_i = 1 \land \alpha_{(i,i+1)}^{\text{break}} = 0 \)), and the first part of a daily rest period (\( \alpha_{i}^{\text{prest}} = 1 \) and \( \text{helpPartialRest} = 0 \)) may take place as well. Note that a reduced daily rest period and a second part of a daily rest period are both considered in the duration of a daily rest period taken at stop \( i + 1 \), \( \Delta_{i+1} \).
2.6. Transformation into a driver schedule

Algorithm 1 Computing a Driver Schedule

**Input:** model solution

**Output:** a list of driver activities

```
1: // Initialize
2: restCarryover<sub>1</sub> ← 0
3: restCarryover<sub>2</sub> ← 0
4: helpPartialRest ← 0
5: ptwr ← Time at the start of the schedule
6: duration ← 0
7: 
8: // Determine activities in the first vertex
9: // ———————————————————————————————
10: if \( \alpha^*_{rest}\) = 1 then
11:    duration ← \( \Delta^*_{rest}\)
12:    addActivity((0, 1), rest, duration)
13:    ptwr ← ptwr + duration
14: else if \( \alpha^*_{break}\) = 1 then
15:    if \( \overline{hp\overline{b}}\) = 1 then
16:       duration ← max(0, 30 − \( \overline{ubl}\))
17:    else
18:       duration ← max(0, 45 − \( \overline{ubl}\))
19:    end if
20:    addActivity((0, 1), rest, duration)
21:    ptwr ← ptwr + duration
22: else if \( \alpha^*_{pbreak}\) = 1 then
23:    duration ← max(0, 15 − \( \overline{ubl}\))
24:    addActivity((0, 1), rest, duration)
25:    ptwr ← ptwr + duration
26: else if \( \alpha^*_{prest}\) = 1 then
27:    duration ← max(0, 180 − \( \overline{urt}\))
28:    addActivity((0, 1), rest, duration)
29:    ptwr ← ptwr + duration
30: else if \( \alpha^*_{prest}\) = 1 then
31:    duration ← max(0, 180 − \( \overline{urt}\))
32:    addActivity((0, 1), rest, duration)
33:    ptwr ← ptwr + duration
```
2. Scheduling of driving times, breaks and rest periods

31: \textbf{end if}
32: if \( \Delta_0 \) \( \ast \) \( \text{wait} \) > 0 then
33: \( \text{duration} \leftarrow \Delta_0 \) \( \ast \) \( \text{wait} \)
34: \( \text{addActivity} \((0, 1), \text{wait}, \text{duration})\)
35: \( \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \)
36: \textbf{end if}
37:
38: /// ———————————————————————————————
39: /// Calculate activities "between" customer locations \( i \) and \( i+1 \)
40: /// ———————————————————————————————
41: \textbf{for} \( i = 0 \) \textbf{to} \( n - 1 \) \textbf{do}
42: /// Use driving time left until next break or rest period to partially or
43: /// completely traverse the arc \((i, i+1)\).
44: if \( L_{i}^{dt} > 0 \) then
45: if \( \Delta_{(i,i+1)}^{\text{drive}} > 0 \) then
46: \( \text{duration} \leftarrow \min \left( L_{i}^{dt}, \Delta_{(i,i+1)}^{\text{drive}} \right) \)
47: \( \text{addActivity} \((0, 1), \text{drive}, \text{duration})\)
48: \( \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \)
49: \textbf{end if}
50: \textbf{end if}
51:
52: /// No early daily rest period is planned as first
53: /// resting activity on arc \((i, i+1)\).
54: if \( \mu_{(i,i+1)} \ast \text{earlydr} \) = 0 then
55: /// The distance between \( i \) and \( i+1 \) is greater than the
56: /// driving time left until the next break or rest period.
57: if \( \Delta_{(i,i+1)}^{\text{drive}} > L_{i}^{dt} \) then
58: /// Take a break if the daily driving time left is greater than
59: /// the driving time left until the next break. Afterwards, continue
60: /// driving until the daily driving time reaches its limit or customer
61: /// \( i+1 \) is reached.
2.6. Transformation into a driver schedule

\[
\begin{align*}
\text{if } L_i^{\text{ddt}} > L_i^{\text{dt}} \text{ then} \\
\quad \text{if } l_i^{\text{break}} = 1 \text{ then} \\
\quad \quad \text{duration} \leftarrow 30 \\
\quad \quad \text{else} \\
\quad \quad \quad \text{if } \alpha_{i+1}^{\text{prest}} = 1 \land \Delta_{(i,i+1)}^{\text{drive}} < L_i^{\text{ddt}} \land \mu_{(i,i+1)}^{\text{extd}} = 0 \land \mu_{i+1}^{\text{extd}} = 0 \text{ then} \\
\quad \quad \quad \quad \text{duration} \leftarrow 180 \\
\quad \quad \quad \quad \text{helpPartialRest} \leftarrow 1 \\
\quad \quad \quad \text{else} \\
\quad \quad \quad \quad \text{duration} \leftarrow 45 \\
\quad \quad \quad \text{end if} \\
\quad \quad \text{end if} \\
\quad \text{addActivity}((i, i+1), \text{rest}, \text{duration}) \\
\quad \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \\
\quad \text{duration} \leftarrow \min \left( L_i^{\text{ddt}} - L_i^{\text{dt}}, \Delta_{(i,i+1)}^{\text{drive}} - L_i^{\text{dt}} \right) \\
\quad \text{addActivity}((i, i+1), \text{drive}, \text{duration}) \\
\quad \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \\
\text{end if}
\end{align*}
\]

// If a driving time extension of type 1 is used, take an additional
// break or first partial daily rest period.
\[
\begin{align*}
\text{if } \mu_{(i,i+1)}^{\text{extd}} = 1 \text{ then} \\
\quad \text{if } \alpha_{i+1}^{\text{prest}} = 1 \land \mu_{i+1}^{\text{extd}} = 0 \land \Delta_{(i,i+1)}^{\text{rest}} = 0 \text{ then} \\
\quad \quad \text{duration} \leftarrow 180 \\
\quad \quad \text{helpPartialRest} \leftarrow 1 \\
\quad \text{else} \\
\quad \quad \text{duration} \leftarrow 45 \\
\quad \text{end if} \\
\quad \text{addActivity}((i, i+1), \text{rest}, \text{duration}) \\
\quad \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \\
\end{align*}
\]

// If the time suffices, then 60 minutes of driving follow, otherwise
// the remaining time left until the next rest period is exploited.
\[
\begin{align*}
\text{if } \lambda_i = 1 \text{ then}
\end{align*}
\]
94: duration ← 60
95: else
96: if \( l_i = 1 \) then
97: duration ← \( L_i - \delta_{\delta dt} - 30 \)
98: else
99: duration ← \( L_i - \delta_{\delta dt} - 45 \)
100: end if
101: if \( \delta_{\delta dt} > L_i \) then
102: duration ← duration − 45
103: end if
104: duration ← \( \min(\text{duration}, \Delta_{\delta_{\delta dr}}(i,i+1) - \delta_{\delta dt}) \)
105: addActivity((i, i + 1), drive, duration)
106: ptwr ← ptwr + duration
107: end if
108: end if
109: end if
110: // ———————————————————————————————
111: // If at least one daily rest period should be made on arc (i, i + 1),
112: // schedule now the first daily rest period.
113: // ———————————————————————————————
114: restCarryover\(_1\) ← \( \Delta_{\delta_{\delta dr}}(i,i+1) - A_{\delta_{\delta dr}}(i,i+1) \cdot 660 + 120 \cdot \mu_{\delta_{\delta dr}}(i,i+1) + 120 \cdot l_i \)
115: if \( A_{\delta_{\delta dr}}(i,i+1) \geq 1 \) then
116: // In case that \( E_{i+1} \) is equal to 540 and only one rest period is taken
117: // between stop \( i \) and \( i + 1 \), the rest period can be postponed to the
118: // subsequent vertex. Only take a daily rest period in the opposite case.
119: if \( E_{i+1} < 540 \lor A_{\delta_{\delta dr}}(i,i+1) > 1 \) then
120: if \( \mu_{\delta_{\delta dr}}(i,i+1) = A_{\delta_{\delta dr}}(i,i+1) \lor l_i = 1 \lor l_i \) then
121: duration ← 540
122: end if
123: end if
124: end if
125: else
126: if \( \alpha_{i+1} = 0 \lor A_{\delta_{\delta dr}}(i,i+1) = 1 \) then
127: duration ← 660
128: end if
129: end if
2.6. Transformation into a driver schedule

\[ \text{duration} \leftarrow \text{duration} + \text{restCarryover}_1 \]
\[ \text{end if} \]
\[ \text{addActivity}((i, i + 1), \text{rest}, \text{duration}) \]
\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]
\[ \text{end if} \]
\[ \text{end if} \]

// Plan driver activities between the first and the last daily rest period
// on arc (i, i + 1)

for \( k = \mathcal{A}_{\text{rest}}^{\ast}(i, i+1) \) to 2 do

\[ \text{duration} \leftarrow 270 \]
\[ \text{addActivity}((i, i + 1), \text{drive}, \text{duration}) \]
\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

if \( k > 2 \lor \mu_{\text{earlydr}}^{\ast}\left(i,i+1\right) = 0 \) then

\[ \text{duration} \leftarrow 45 \]
\[ \text{addActivity}((i, i + 1), \text{rest}, \text{duration}) \]
\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]
\[ \text{duration} \leftarrow 270 \]
\[ \text{addActivity}((i, i + 1), \text{drive}, \text{duration}) \]
\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

// Schedule extended driving times of type 2 as late as possible.

if \( k \leq \mu_{\text{extd}}^{\ast}(i,i+1) + 1 \) then

\[ \text{duration} = 45 \]
\[ \text{addActivity}((i, i + 1), \text{rest}, \text{duration}) \]
\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

\[ \text{duration} \leftarrow 60 \]
\[ \text{addActivity}((i, i + 1), \text{drive}, \text{duration}) \]
\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

end if

if \( k > 2 \) then

// Schedule reduced daily rest periods as late as possible.
2. Scheduling of driving times, breaks and rest periods

// In case that \( \ell_i \) = 1, one reduced daily rest period has to be the first daily rest period on this arc.

if \( k \leq \mu^{\text{redrest}}_{(i,i+1)} - \ell_i \) then
  duration ← 540
  addActivity((i, i + 1), rest, duration)
  ptwr ← ptwr + duration
else
  duration ← 660
  addActivity((i, i + 1), rest, duration)
  ptwr ← ptwr + duration
end if

end if

end for

// Plan last daily rest period in case that more than one daily rest period is taken on arc \((i, i+1)\).

if \( A^{\text{rest}}_{(i,i+1)} \geq 2 \) then
  if \( E^{\text{add}}_{i+1} < 540 \) then
    if \( \mu^{\text{redrest}}_{(i,i+1)} \geq 1 + \ell_i \) then
      duration ← 540
    else
      duration ← 660
    end if
  end if
else
  if \( a^{\text{rest}}_{i+1} = 0 \) then
    duration ← duration + restCarryover_1
  end if
  addActivity((i, i + 1), rest, duration)
  ptwr ← ptwr + duration
end if

end if
2.6. Transformation into a driver schedule

1. Plan driver activities after the last daily rest period on arc $(i, i+1)$.

if $\Delta_{rest}(i, i+1) \geq 1$ then
  if $E_{i+1} < 540$ then
    duration $\leftarrow \min \left(270, 540 - \Delta_{ddt}(i+1)\right)$
    $addActivity((i, i+1), drive, duration)$
    $ptwr \leftarrow ptwr + duration$
  
  if $E_{i+1} < 270$ then
    if $\alpha_{i+1} = 1 \land \mu_{i+1} = 0 \land \mu_{extd}(i, i+1) = 0$ then
      duration $\leftarrow 180$
      helpPartialRest $\leftarrow 1$
    else
      duration $\leftarrow 45$
  end if
  end if
  $addActivity((i, i+1), rest, duration)$
  $ptwr \leftarrow ptwr + duration$
  if $\mu_{extd}(i, i+1) = 0$ then
    duration $\leftarrow 270 - \Delta_{ddt}(i+1)$
    $addActivity((i, i+1), drive, duration)$
    $ptwr \leftarrow ptwr + duration$
  else
    duration $\leftarrow 270$
    $addActivity((i, i+1), drive, duration)$
    $ptwr \leftarrow ptwr + duration$
    if $\alpha_{i+1} = 1 \land \mu_{i+1} = 0$ then
      duration $\leftarrow 180$
      helpPartialRest $\leftarrow 1$
    else
      duration $\leftarrow 45$
    end if
  end if
  $addActivity((i, i+1), rest, duration)$
  $ptwr \leftarrow ptwr + duration$
2. Scheduling of driving times, breaks and rest periods

\[ \text{duration} \leftarrow 60 - \sum_{i}^{} ddt_i \]

if \( \text{duration} > 0 \) then

\[ \text{addActivity} ((i, i+1), \text{drive, duration}) \]

\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

end if

end if

end if

end if

end if

end if

end if

end if

// ———————————————————————————————
// Plan driver activities at arrival at customer \( i + 1 \).
// ———————————————————————————————

// If a daily rest period is planned for arc \((i, i+1)\) that can be postponed
// to the subsequent vertex, do it. If a daily rest period is also scheduled
// at customer location \( i + 1 \), unite the two daily rest periods.

if \( E_{i+1} = 540 \land A_{(i,i+1)} \geq 1 \) then

if \( \mu_{(i,i+1)} \geq 1 + l_i \lor \left[ A_{(i,i+1)} = 1 \land \left( \mu_{(i,i+1)} = 1 \lor l_i = 1 \right) \right] \) then

\[ \text{restCarryover}_2 \leftarrow 540 \]

else

\[ \text{restCarryover}_2 \leftarrow 660 \]

end if

end if

if \( A_{i+1}^\text{rest} = 1 \) then

\[ \text{duration} \leftarrow \Delta_{i+1}^\text{rest} + \text{restCarryover}_1 + \text{restCarryover}_2 \]

\[ \text{addActivity} ((i, i+1), \text{rest, duration}) \]

\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

else if \( \text{restCarryover}_2 > 0 \) then

\[ \text{duration} \leftarrow \text{restCarryover}_2 + \text{restCarryover}_1 \]

\[ \text{addActivity} ((i, i+1), \text{rest, duration}) \]

\[ \text{ptwr} \leftarrow \text{ptwr} + \text{duration} \]

else if \( A_{(i,i+1)} \geq 1 \land E_{i+1} = 270 \land E_{i+1} \leq 270 \) then

if \( l_i = 1 \land A_{(i,i+1)} = 0 \) then

\[ \text{duration} \leftarrow 30 \]
2.6. Transformation into a driver schedule

270: else
271: duration ← 45
272: end if
273: addActivity ((i, i + 1), rest, duration)
274: ptwr ← ptwr + duration
275: end if
276: if \( \alpha_{i+1}^{break} = 1 \) then
277: if \( l_i^{break} = 1 \land \alpha_{(i,i+1)}^{break} = 0 \) then
278: duration ← 30
279: else
280: duration ← 45
281: end if
282: addActivity ((i, i + 1), rest, duration)
283: ptwr ← ptwr + duration
284: else if \( \alpha_{i+1}^{break} = 1 \) then
285: duration ← 15
286: addActivity ((i, i + 1), rest, duration)
287: ptwr ← ptwr + duration
288: else if \( \alpha_{i+1}^{prest} = 1 \land helpPartialRest = 0 \) then
289: duration ← 180
290: addActivity ((i, i + 1), rest, duration)
291: ptwr ← ptwr + duration
292: end if
293: if \( \Delta_{i+1}^{wait} > 0 \) then
294: duration ← \( \Delta_{i+1}^{wait} \)
295: addActivity ((i, i + 1), wait, duration)
296: ptwr ← ptwr + duration
297: end if
298: if \( i < r - 1 \land \Delta_{i+1}^{service} > 0 \) then
299: duration ← \( \Delta_{i+1}^{service} \)
300: addActivity ((i, i + 1), work, duration)
301: ptwr ← ptwr + duration
302: end if
303: i ← i + 1
304: end for
2.7. Numerical experiments - Part 1

The MILP models described in the previous section were implemented in Java (Java 8, 64 bit) and were solved with CPLEX 12.6 (64 bit) with ILOG CPLEX Concert Technology. The test runs were performed on an Intel Core i5 2500K with 8 GB RAM (DDR3-10700 (667 MHz)) running Windows 7 Professional Service Pack 1, 64 bit.

If the number of time windows per customer location (vertices $1$ to $r - 2$) is denoted by $z$ and the final destination is equipped with a "time window" that equals the planning horizon, the first MILP submodel (for minimizing lateness along with completion time) with optional rules has

- $28r + rz - 2z - 7$ binary variables,
- $4(r - 1)$ integer (non-binary) variables,
- $11r - 2$ continuous variables and
- $147r - 49$ constraints

where $r$ denotes the number of customer locations to be visited (including start and end position).

In the following, test instances are derived from real routes that were obtained from a transport company. Then the run times for two possible solution processes for the model with optional rules are compared. If no specific optimization step is addressed, the run times comprise the CPLEX times for all optimization steps involved and are given as wall clock time\textsuperscript{42}. Afterwards, run times are analyzed depending on the number of stops and the number and size of time windows. The section concludes with the analysis of the influence of the optional rules on the run time as well as on lateness and overall schedule duration.

2.7.1. Test instances

Test instances were derived from real data provided by a German haulage company that operates a fleet of vehicles in Europe. The haulage company was a partner in the research project Dynaserv which aimed at offering decision support for dynamic tasks of dispatchers in transport companies by the integration of online data. The underlying database of the prototype developed during the project comprised telematics data of the vehicles of the haulage company as well as arrival times at customer locations initially planned by the dispatchers in the order management system. Routes and driving times (without rest periods and breaks) between customer locations were first calculated using the routing algorithm of the prototype. The routing algorithm can differentiate between different road types when determining the fastest routes. Still, other criteria exist that are important in practice. Drivers were, for example, instructed for chosen highway segments to use toll free

\textsuperscript{42} Wall clock time: total physical time elapsed
state roads that were very close to the original route instead to save money. Additionally, driver preferences and the suitability of roads for trucks play a role. Therefore, for each route, support points\textsuperscript{43} were manually added until the route computed by the routing algorithm matched the route actually chosen by the driver. Figure 2.11 shows how a support point is added.

![Figure 2.11.: Fitting the computed route to the driver's route](image)

A planning horizon of one week was considered. In all test cases, drivers start their tours on Monday after a weekly rest period. Varying starting times were taken into account. To be able to analyze the influence of the number of time windows on the run time, cases with one, two or three alternative time windows were generated. Exact time windows at customer locations were not available as they were recorded in customer systems and not transferred to the database of the prototype. Instead, planned arrival times extracted from the transport management software were used as a basis for the test instances. From process analysis it is known that arrival times were planned to be sooner rather than late. Therefore, for the cases with one or two time windows, the start of the first time window at a customer location was set to be equal to the planned arrival time. For the case with three alternative time windows, the start of the intermediate time window was set to be equal to the planned arrival time. Time window lengths of 0, 30, 60, 120 and 600 minutes were analyzed. The time between the end of a time window and the start of the subsequent time window was set to 120 minutes except for the instances with 600 minutes time windows. Here, a time interval of 840 minutes was chosen. These instances represent daily opening

\textsuperscript{43} Support points, similar to the customer locations, have to be traversed by the route computed by the routing algorithm. They were used as a tool to fit the computed routes to the drivers' routes for which position data was available.
hours. Table 2.1 gives an overview of the underlying vehicle routes of the base instances. The values indicated in the last two columns correspond to the real distances traveled and the total real driving time.

<table>
<thead>
<tr>
<th>base instance/country</th>
<th># stops including start and end</th>
<th>countries stops</th>
<th>total distance in km</th>
<th>overall driving duration in h</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>HU-DE-DE-ES</td>
<td>2914</td>
<td>36.85</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>ES-DE-DE-IT</td>
<td>3391</td>
<td>42.48</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>DE-DE-DE-ES-ES-DE</td>
<td>3653</td>
<td>46.95</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>HU-DE-DE-IT-IT-DE</td>
<td>2831</td>
<td>36.45</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>DE-DE-DE-DE-IT-IT</td>
<td>2944</td>
<td>37.47</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>ES-FR-FR-FR-FR-FR-DE-DE</td>
<td>2269</td>
<td>30.32</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>DE-DE-IT-IT-IT-DE-DE-HU-HU</td>
<td>3142</td>
<td>39.77</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>IT-IT-IT-IT-DE-DE-HU-IT</td>
<td>3019</td>
<td>38.17</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>DK-DK-DK-DK-FR-FR-DK</td>
<td>3436</td>
<td>43.77</td>
</tr>
</tbody>
</table>

Table 2.1.: Characteristics of the test instances

For each base instance, 15 test instances were obtained by varying the number and length of time windows as described above. In total, 225 test instances were generated. Table 2.2 shows the number of variables and constraints for the test instances with three alternative time windows depending on the number of stops.

<table>
<thead>
<tr>
<th># stops (incl. start and end)</th>
<th># binary variables</th>
<th># integer (non-binary variables)</th>
<th># continuous variables</th>
<th># constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>111</td>
<td>12</td>
<td>42</td>
<td>539</td>
</tr>
<tr>
<td>5</td>
<td>142</td>
<td>16</td>
<td>53</td>
<td>686</td>
</tr>
<tr>
<td>6</td>
<td>173</td>
<td>20</td>
<td>64</td>
<td>833</td>
</tr>
<tr>
<td>7</td>
<td>204</td>
<td>24</td>
<td>75</td>
<td>980</td>
</tr>
<tr>
<td>8</td>
<td>235</td>
<td>28</td>
<td>86</td>
<td>1127</td>
</tr>
<tr>
<td>9</td>
<td>266</td>
<td>32</td>
<td>97</td>
<td>1274</td>
</tr>
<tr>
<td>10</td>
<td>297</td>
<td>36</td>
<td>108</td>
<td>1421</td>
</tr>
<tr>
<td>11</td>
<td>328</td>
<td>40</td>
<td>119</td>
<td>1568</td>
</tr>
<tr>
<td>12</td>
<td>359</td>
<td>44</td>
<td>130</td>
<td>1715</td>
</tr>
</tbody>
</table>

Table 2.2.: Total number of variables and constraints in the MILP model with optional rules (3 alternative time windows)
2.7.2. The solution of the model without optional rules as upper cutoff

In order to reduce the CPLEX run time, the possibility to set an upper cutoff for the optimal solution value was examined. Run times for the model without optional rules were known to be significantly shorter (see Section 2.7.3). Therefore, we used the optimal solution value obtained by solving the first submodel with a solution space restricted to possibilities that do not include optional rules as an upper cutoff. Figure 2.12 gives an overview of the solution processes with and without upper cutoff. Tables 2.3 and 2.4 display the run times for all test instances cumulated over 2 or 3 steps, respectively.

![Solution processes with and without an upper cutoff](image)

Figure 2.12.: Solution processes with and without an upper cutoff

All instances were solved to optimality for both solution processes. The run times varied between 0.03 seconds and 79.748 seconds without additional run for an upper cutoff. With the upper cutoff implemented, the runs only took between 0.03 seconds and 10.935 seconds. On average, the run time was almost 38% less when an upper cutoff was used. The two instances derived from base instance 15 with a time window length of 600 minutes and one or two time windows show the greatest improvements and are crucial for the reduced average run time. This can be identified clearly in Figure 2.13.
Table 2.3.: Run times in seconds for the MILP model without additional step for an upper cutoff
<table>
<thead>
<tr>
<th>base inst./ route</th>
<th># stops</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.064</td>
<td>0.141</td>
<td>0.047</td>
<td>0.063</td>
<td>0.078</td>
<td>0.393</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.109</td>
<td>0.203</td>
<td>0.218</td>
<td>0.109</td>
<td>0.109</td>
<td>0.748</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.063</td>
<td>0.062</td>
<td>0.077</td>
<td>0.079</td>
<td>0.125</td>
<td>0.406</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.110</td>
<td>0.172</td>
<td>0.124</td>
<td>0.094</td>
<td>0.155</td>
<td>0.655</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.126</td>
<td>0.095</td>
<td>0.095</td>
<td>0.095</td>
<td>0.125</td>
<td>0.536</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.126</td>
<td>0.140</td>
<td>0.125</td>
<td>0.140</td>
<td>0.188</td>
<td>0.719</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.311</td>
<td>0.295</td>
<td>0.375</td>
<td>0.281</td>
<td>0.265</td>
<td>1.527</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.125</td>
<td>0.123</td>
<td>0.141</td>
<td>0.140</td>
<td>0.312</td>
<td>0.841</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.125</td>
<td>0.125</td>
<td>0.188</td>
<td>0.126</td>
<td>0.110</td>
<td>0.674</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.764</td>
<td>0.280</td>
<td>0.405</td>
<td>0.437</td>
<td>0.328</td>
<td>2.214</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.436</td>
<td>0.687</td>
<td>0.344</td>
<td>0.328</td>
<td>0.937</td>
<td>2.732</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0.959</td>
<td>0.934</td>
<td>0.531</td>
<td>0.545</td>
<td>0.907</td>
<td>3.509</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>1.009</td>
<td>1.077</td>
<td>0.904</td>
<td>0.609</td>
<td>1.575</td>
<td>4.974</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>1.997</td>
<td>2.168</td>
<td>1.826</td>
<td>1.685</td>
<td>1.576</td>
<td>9.252</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>1.529</td>
<td>1.637</td>
<td>0.984</td>
<td>2.059</td>
<td>3.962</td>
<td>19.061</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7.286</td>
<td>8.139</td>
<td>15.274</td>
<td>6.790</td>
<td>10.752</td>
<td>48.241</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.077</td>
<td>0.062</td>
<td>0.062</td>
<td>0.079</td>
<td>0.063</td>
<td>0.343</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.108</td>
<td>0.077</td>
<td>0.079</td>
<td>0.094</td>
<td>0.126</td>
<td>0.484</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.078</td>
<td>0.062</td>
<td>0.062</td>
<td>0.095</td>
<td>0.157</td>
<td>0.454</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.172</td>
<td>0.203</td>
<td>0.186</td>
<td>0.172</td>
<td>0.203</td>
<td>0.936</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.109</td>
<td>0.717</td>
<td>0.094</td>
<td>0.141</td>
<td>0.203</td>
<td>1.264</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.154</td>
<td>0.218</td>
<td>0.265</td>
<td>0.343</td>
<td>0.173</td>
<td>1.153</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.266</td>
<td>0.311</td>
<td>0.250</td>
<td>0.203</td>
<td>0.219</td>
<td>1.249</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.172</td>
<td>0.171</td>
<td>0.155</td>
<td>0.124</td>
<td>0.172</td>
<td>0.794</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.156</td>
<td>0.125</td>
<td>0.156</td>
<td>0.234</td>
<td>0.218</td>
<td>0.889</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.576</td>
<td>0.375</td>
<td>0.375</td>
<td>0.437</td>
<td>0.321</td>
<td>2.075</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0.843</td>
<td>0.764</td>
<td>0.437</td>
<td>0.547</td>
<td>0.374</td>
<td>2.965</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>1.265</td>
<td>0.951</td>
<td>0.779</td>
<td>1.091</td>
<td>0.905</td>
<td>4.991</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0.949</td>
<td>0.514</td>
<td>0.968</td>
<td>0.749</td>
<td>0.749</td>
<td>4.154</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>2.137</td>
<td>1.576</td>
<td>1.389</td>
<td>0.717</td>
<td>2.324</td>
<td>8.143</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.077</td>
<td>0.109</td>
<td>0.078</td>
<td>0.095</td>
<td>0.030</td>
<td>0.389</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.110</td>
<td>0.141</td>
<td>0.077</td>
<td>0.093</td>
<td>0.265</td>
<td>0.686</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.078</td>
<td>0.063</td>
<td>0.094</td>
<td>0.110</td>
<td>0.359</td>
<td>0.704</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.266</td>
<td>0.311</td>
<td>0.219</td>
<td>0.141</td>
<td>0.110</td>
<td>0.909</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.375</td>
<td>0.156</td>
<td>0.126</td>
<td>0.142</td>
<td>0.110</td>
<td>0.909</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.421</td>
<td>0.437</td>
<td>0.578</td>
<td>0.249</td>
<td>0.265</td>
<td>1.950</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.344</td>
<td>0.234</td>
<td>0.281</td>
<td>0.373</td>
<td>0.406</td>
<td>1.638</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2.058</td>
<td>0.296</td>
<td>0.186</td>
<td>0.422</td>
<td>0.203</td>
<td>3.165</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.156</td>
<td>0.140</td>
<td>0.188</td>
<td>0.204</td>
<td>0.390</td>
<td>1.078</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0.655</td>
<td>0.422</td>
<td>0.469</td>
<td>0.453</td>
<td>0.701</td>
<td>2.700</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>1.996</td>
<td>1.435</td>
<td>1.030</td>
<td>0.703</td>
<td>1.686</td>
<td>6.850</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0.889</td>
<td>1.373</td>
<td>1.403</td>
<td>1.404</td>
<td>1.139</td>
<td>6.208</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>1.014</td>
<td>1.201</td>
<td>1.420</td>
<td>1.139</td>
<td>3.416</td>
<td>8.190</td>
</tr>
<tr>
<td></td>
<td>Σ</td>
<td>14.492</td>
<td>11.078</td>
<td>10.641</td>
<td>11.315</td>
<td>27.097</td>
<td>74.623</td>
</tr>
</tbody>
</table>

Table 2.4.: Run times in seconds for the MILP model with additional step for an upper cutoff
Figure 2.13.: Run times for the MILP model with optional rules
For further analysis of the run time, the solution process with three steps, that means with one additional step to determine an upper cutoff, was chosen.

### 2.7.3. Analysis

Figure 2.14 depicts the average run times of steps 1 to 3 depending on the number of customer vertices. As expected, the run time increases with the number of customer vertices. It is not surprising that the run time of the second step grows strongest, thus increasing the proportion of the second step on the overall run time from almost 42% (4 stops) to more than 71% (12 stops).

![Figure 2.14: Average run time depending on the number of vertices](image)

The influence of the number of time windows on the run time

Figure 2.15 shows the influence of the number of time windows on the average run time depending on the number of vertices. In the following examples it clearly can be seen that the number of time windows per stop as well as the number of vertices are important for the run time.

Example 1: 10 stops, 2 time windows:

- Total number of variables: 433.
• Number of constraints: 1421.
• Average run time: 0.908 s

Example 2: 11 stops, 2 time windows:
• Total number of variables: 478. Increase compared to Example 1: 10%.
• Number of constraints: 1568. Increase compared to Example 1: 10%.
• Average run time: 1.629 s. Increase compared to Example 1: 79%.

Example 3: 10 stops, 3 time windows:
• Total number of variables: 441. Increase compared to Example 1: 2%.
• Number of constraints: 1421. Increase compared to Example 1: 0%.
• Average run time: 1.638 s. Increase compared to Example 1: 80%.

![Graph showing run time vs. number of stops and time windows](image)

Figure 2.15.: The influence of the number of time windows (TW) on the run time

**The influence of the time window length on the run time**

The three diagrams in Figure 2.16 show the average run times depending on the number of stops and the time window length separately for each number of alternative time windows per stop.
Figure 2.16.: The influence of the time window length on the run time
We can see from Figure 2.16 that there are significantly long run times in some of the instances with a time window length of 10 h compared to the other instances.

Furthermore, there is one test instance (there is only one base instance that has 12 stops) with a time window length of 120 minutes that has a comparably long run time and another one with a time window length of 60 minutes. Test instances with time window lengths of 30 minutes or exact arrival times performed best.

If we consider the cumulative view for all numbers of alternative time windows (Figure 2.17), in some cases, we can see significant decreasing average run times if a time window with a length of 30 minutes instead of an exact arrival time is possible. In others, a slight increase can be recognized.

![Figure 2.17: The influence of the time window length on the run time](image)

**The influence of the optional rules**

In the following, we examine the influence of the optional rules on the run time and the reachability of time windows. For the case without optional rules, constraints (2.5.187) - (2.5.202) are included. For the case with optional rules, the solution process with upper cutoff is chosen (see Figure 2.12). Table 2.5 shows the cumulated run time of both steps for each of the 225 test instances for the model without optional rules. Figure 2.18 compares the average run time for each base instance with the run time of the MILP model with optional rules (see also Table 2.4 on page 105). Ignoring the optional rules, the average run time of the test instances per base instance could be reduced by 37% (base instance 1, 4 stops) up to 81% (base instance 15, 12 stops). This means that the optional rules have a strong influence on the overall run time as expected.
<table>
<thead>
<tr>
<th>base inst./ route</th>
<th># stops</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.032</td>
<td>0.046</td>
<td>0.046</td>
<td>0.047</td>
<td>0.032</td>
<td>0.203</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.031</td>
<td>0.031</td>
<td>0.046</td>
<td>0.046</td>
<td>0.077</td>
<td>0.231</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.048</td>
<td>0.032</td>
<td>0.046</td>
<td>0.031</td>
<td>0.046</td>
<td>0.203</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.047</td>
<td>0.062</td>
<td>0.032</td>
<td>0.079</td>
<td>0.063</td>
<td>0.283</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.063</td>
<td>0.031</td>
<td>0.046</td>
<td>0.031</td>
<td>0.077</td>
<td>0.248</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.094</td>
<td>0.031</td>
<td>0.047</td>
<td>0.063</td>
<td>0.047</td>
<td>0.282</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0.094</td>
<td>0.077</td>
<td>0.094</td>
<td>0.094</td>
<td>0.061</td>
<td>0.420</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>0.062</td>
<td>0.062</td>
<td>0.063</td>
<td>0.062</td>
<td>0.094</td>
<td>0.343</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>0.062</td>
<td>0.032</td>
<td>0.079</td>
<td>0.030</td>
<td>0.078</td>
<td>0.281</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>0.078</td>
<td>0.110</td>
<td>0.124</td>
<td>0.141</td>
<td>0.125</td>
<td>0.578</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>0.125</td>
<td>0.282</td>
<td>0.234</td>
<td>0.219</td>
<td>0.281</td>
<td>1.141</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>0.484</td>
<td>0.390</td>
<td>0.608</td>
<td>0.468</td>
<td>0.406</td>
<td>2.256</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0.266</td>
<td>0.296</td>
<td>0.624</td>
<td>0.343</td>
<td>0.250</td>
<td>1.779</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>0.405</td>
<td>0.437</td>
<td>0.327</td>
<td>0.235</td>
<td>0.872</td>
<td>2.276</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>0.469</td>
<td>0.795</td>
<td>0.935</td>
<td>0.531</td>
<td>0.249</td>
<td>2.979</td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>2.360</td>
<td>2.714</td>
<td>3.351</td>
<td>2.420</td>
<td>2.758</td>
<td>13.603</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.063</td>
<td>0.046</td>
<td>0.032</td>
<td>0.046</td>
<td>0.031</td>
<td>0.218</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.031</td>
<td>0.047</td>
<td>0.062</td>
<td>0.078</td>
<td>0.062</td>
<td>0.280</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.047</td>
<td>0.046</td>
<td>0.032</td>
<td>0.046</td>
<td>0.063</td>
<td>0.234</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.053</td>
<td>0.047</td>
<td>0.093</td>
<td>0.079</td>
<td>0.093</td>
<td>0.375</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.047</td>
<td>0.078</td>
<td>0.047</td>
<td>0.062</td>
<td>0.093</td>
<td>0.327</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.079</td>
<td>0.062</td>
<td>0.078</td>
<td>0.109</td>
<td>0.063</td>
<td>0.391</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0.093</td>
<td>0.110</td>
<td>0.125</td>
<td>0.063</td>
<td>0.078</td>
<td>0.469</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>0.078</td>
<td>0.047</td>
<td>0.047</td>
<td>0.047</td>
<td>0.110</td>
<td>0.329</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>0.078</td>
<td>0.062</td>
<td>0.063</td>
<td>0.136</td>
<td>0.109</td>
<td>0.468</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>0.140</td>
<td>0.218</td>
<td>0.203</td>
<td>0.125</td>
<td>0.187</td>
<td>0.873</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>0.297</td>
<td>0.281</td>
<td>0.624</td>
<td>0.203</td>
<td>0.291</td>
<td>2.686</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>0.577</td>
<td>0.515</td>
<td>0.421</td>
<td>0.593</td>
<td>0.374</td>
<td>2.480</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0.297</td>
<td>0.296</td>
<td>0.343</td>
<td>0.187</td>
<td>0.343</td>
<td>1.466</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>0.234</td>
<td>0.312</td>
<td>0.373</td>
<td>0.344</td>
<td>0.483</td>
<td>1.746</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>1.029</td>
<td>1.061</td>
<td>0.748</td>
<td>0.780</td>
<td>0.375</td>
<td>3.993</td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>3.153</td>
<td>3.228</td>
<td>3.291</td>
<td>2.918</td>
<td>2.745</td>
<td>15.335</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.063</td>
<td>0.094</td>
<td>0.047</td>
<td>0.031</td>
<td>0.047</td>
<td>0.282</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.063</td>
<td>0.063</td>
<td>0.031</td>
<td>0.079</td>
<td>0.063</td>
<td>0.299</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0.064</td>
<td>0.031</td>
<td>0.016</td>
<td>0.079</td>
<td>0.124</td>
<td>0.296</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.079</td>
<td>0.079</td>
<td>0.110</td>
<td>0.094</td>
<td>0.094</td>
<td>0.456</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.030</td>
<td>0.062</td>
<td>0.046</td>
<td>0.062</td>
<td>0.046</td>
<td>0.246</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.077</td>
<td>0.141</td>
<td>0.078</td>
<td>0.109</td>
<td>0.110</td>
<td>0.515</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0.157</td>
<td>0.110</td>
<td>0.078</td>
<td>0.062</td>
<td>0.063</td>
<td>0.470</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>1.982</td>
<td>0.171</td>
<td>0.110</td>
<td>0.109</td>
<td>0.391</td>
<td>2.763</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>0.093</td>
<td>0.093</td>
<td>0.094</td>
<td>0.094</td>
<td>0.061</td>
<td>0.435</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>0.219</td>
<td>0.172</td>
<td>0.218</td>
<td>0.312</td>
<td>0.312</td>
<td>1.233</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>0.639</td>
<td>0.515</td>
<td>0.156</td>
<td>0.531</td>
<td>0.312</td>
<td>2.153</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>0.655</td>
<td>0.625</td>
<td>0.842</td>
<td>0.764</td>
<td>0.515</td>
<td>3.401</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0.515</td>
<td>0.436</td>
<td>0.983</td>
<td>0.297</td>
<td>0.811</td>
<td>3.042</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>0.780</td>
<td>0.780</td>
<td>0.951</td>
<td>0.530</td>
<td>1.154</td>
<td>4.195</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>0.638</td>
<td>0.484</td>
<td>0.889</td>
<td>0.638</td>
<td>1.264</td>
<td>3.913</td>
</tr>
</tbody>
</table>

Table 2.5.: Run times in seconds for the MILP model without optional rules
Figure 2.18.: Average run times \textbf{with} and \textbf{without} considering optional rules
Total lateness, on the other hand, is reduced significantly if optional rules are considered. Figure 2.19 shows the average lateness per base instance.

![Graph showing average total lateness with and without considering optional rules](image)

Figure 2.19: Average total lateness with and without considering optional rules

The large difference for base instance 12 is notable. Here, time windows very often cannot be met at all and in some instances lateness at a single customer location of sometimes more than 10 hours cannot be avoided. In practice, re-planning would be necessary. Only in the instances with simulated opening hours and at least two time windows lateness can be avoided, no matter if the optional rules are used or not. When considering the optional rules, the largest share of the lateness reduction is achieved because in some of the instances a complete daily rest period can be left out. In these instances, this has an effect on the lateness at several customer locations. For base instances 1 and 2 no lateness occurred even if optional rules were ignored. For base instances 4 and 9, the average lateness was not reduced when considering the optional rules. In all other cases, a reduction was noticed. Cumulating the lateness over all test instances, a reduction of 55% can be observed by taking into account optional rules.

The influence of the optional rules on the overall schedule duration, that means, the overall time from the start of the schedule until the last stop is reached, should not be neglected as well. Figure 2.20 shows the average schedule duration with and without optional rules per base instance. The computational results show that on average over all test instances the schedule duration is reduced by 5% when optional rules are considered. The average reduction of the schedule duration is nearly 6 hours (357 minutes), this means that the last stop can be reached much earlier. Thus, the start of the weekly rest period can be earlier if optional rules are taken into account. This option is interesting if drivers return home for the weekend. If not, as it is the case in the test instances, they may continue driving and
may reach the next customer earlier. Furthermore, drivers may use part of the additional time to look for a good rest area to stay for the weekend.

![Graph showing average schedule duration with and without considering optional rules.](image)

Figure 2.20.: Average schedule duration with and without considering optional rules

Summing up, it is really worthwhile to consider the optional rules when planning driver schedules. The test scenarios presented so far do not consider unexpected events like for example traffic jams, but such events may extend the travel times significantly. Therefore, it is important to include time buffers, for example, in the driving durations between two consecutive stops and in the loading and unloading times, no matter which planning technique is chosen. This can be done by multiplying the durations with a constant factor that is greater than one and using the result as new estimated duration. Thus, time buffers are obtained that are proportional to the durations. Other techniques that, for example, incorporate road data and the probability of traffic jams for different roads are also possible.
2.8. Myopic algorithm - A heuristic

The MILP models presented in the previous sections to set up a driver schedule were solved using an optimization solver. When integrating optimization software, acquisition and integration costs must not be neglected. A transport company has to weigh the additional costs against the advantages and cost savings achievable.

To measure the added value of the proposed solution method, a myopic heuristic was developed that mimics the manual planning process of an experienced dispatcher that uses sophisticated strategies to plan driver activities. We expect that a dispatcher would consider the routes between customer locations and time windows successively one after another as this still allows a manual evaluation of possibilities and adopt this strategy in the heuristic.

The input for the heuristic is the same as for the models presented: the driver status at the beginning of the planning horizon, the sequence of customer locations and other stops to be visited, driving durations between consecutive stops $i$ and $i + 1$ ($\Delta_{\text{drive}}^{(i,i+1)}$), start and end times of possible time windows at stop $i$ ($TW_{i\text{begin}}$, $TW_{i\text{end}}$) and planned durations for handling activities including loading and/or unloading the vehicle ($\Delta_{i\text{service}}$).

Similar to the naive labeling algorithm proposed by Goel (2009), the driver status is represented by an $n$-tupel. The three-tupel described in Goel (2009) that defines the driver status upon arrival at a stop contains the arrival time, the cumulated driving time since the last (daily or weekly) rest period and the cumulated driving time since the last break or rest period (nonstop driving time). The myopic heuristic, similar to the naive labeling algorithm, only considers at each decision point the driver status and activities concerning the current arc (i.e. decisions about activities that take place between leaving stop $i$ and loading or unloading at stop $i + 1$) and decides for exactly one alternative. Once the plan for an arc is made, the driver status at the subsequent stop is fixed and the algorithm proceeds with planning the next arc.

The myopic heuristic is structured as follows. At first, the driver status is initialized. Then, activities between each pair of successive stops $i$ and $i + 1$ are scheduled sequentially. This is done in three steps. In step one, activities between stops $i$ and $i + 1$ are scheduled. Step two is concerned with the choice of the time window at the next stop $i + 1$. In step three, activities at stop $i + 1$ are scheduled. Steps one to three are repeated until the last stop is reached. Figure 2.21 shows a flowchart of the complete algorithm. In the following, more detailed flowcharts and pseudo code are given for the different steps. In Section 2.8.1, the driver status and its initialization are described. Section 2.8.2 introduces the update algorithm for the driver status which is used in each of the three steps described above. The "first reachable time window" needs to be determined in two of the three steps. The corresponding algorithm is presented in Section 2.8.3. For each pair of consecutive stops $i$ and $i + 1$, Algorithms 5, 6, and 7 are executed one after another to determine the driver schedule.
start

Initialize driver status

stop number = 0

Scheduling activities on arc (i,j+1)
Schedule activities between stops i and i+1.

Choose time window at stop i+1
• Choose a time window and
• plan an additional daily rest period, if necessary.

Modify rest durations and schedule activities at stop i+1
• Reduce daily rest periods planned if it is possible to reduce lateness.
• Plan activities at stop i+1.

stop number = stop number + 1

yes

stop number < n

no

end

Figure 2.21.: Myopic heuristic - flowchart
2.8.1. The driver status

We extend the tuple representing the driver status to be able to additionally consider the optional rules. To this end, the following variables are considered:

- $ptwr$: Time elapsed since the last weekly rest period in minutes.
- $udt$: Uninterrupted driving time since the last break or daily rest period.
- $ddt$: Cumulated driving time since the last daily or weekly rest period.
- $ptr$: Time elapsed since the end of the last daily or weekly rest period.
- $hpb$: Takes the value 1 if a partial break was taken and 0 otherwise.
- $hpr$: 1 if a partial daily rest period has been taken, 0 otherwise.
- $noRed$: Number of reduced daily rest periods taken since the end of the last weekly rest period. (Previous daily rest periods with a duration of less than 11 hours).
- $noExt$: Number of extended daily driving times already taken in this week. If the current daily driving time exceeds 9 hours, this information is included.
- $red$: 1 if the next daily rest period is planned to be a reduced one, 0 otherwise.
- $dte$: 1 if a driving time extension is active (more than 9 hours of daily driving time), 0 otherwise.

The driver status is represented by a 10-tuple

$$driverStatus = (ptwr, udt, ddt, ptr, hpb, hpr, noRed, noExt, red, dte).$$

The corresponding status variables are initialized in Algorithm 2.

**Algorithm 2 Initialize driver status**

1: // Initialize: Set starting driver status
2:    $udt \leftarrow udt$
3:    $ddt \leftarrow ddt$
4:    $ptr \leftarrow ptr$
5:    $ptwr \leftarrow ptwr$
6:    $hpb \leftarrow hpb$
7:    $hpr \leftarrow hpr$
2. Scheduling of driving times, breaks and rest periods

9: \( \text{noRed} \leftarrow \text{noRed} \)
10: \( \text{noExt} \leftarrow \text{noExt} \)
11: \( dte \leftarrow 0 \)
12: \( red \leftarrow 0 \)
13:
14: // If the driving time since the last daily or weekly rest period exceeds 9 hours, a
15: // driving time extension is active.
16:
17: if \( ddt > 540 \land noExt \leq 2 \) then
18: \( dte \leftarrow 1 \)
19: end if
20:
21: // If the time since the last daily or weekly rest period exceeds 13 hours, plan that
22: // the next daily rest period has to be a reduced one.
23:
24: if \( ptr > 780 \land noRed < 3 \) then
25: \( red \leftarrow 1 \)
26: end if

2.8.2. Updating the driver status

Each activity has an activity type and a duration, and when added to the schedule it modifies the driver status. Activity types considered by the myopic heuristic are:

- **rest**: Regular, first or second part of a daily rest period
- **redrest**: Reduced daily rest period
- **drive**: Driving
- **work**: Loading or unloading goods
- **break**: Break
- **wait**: Wait

The method \( \text{scheduleActivity}(<\text{last stop}>, <\text{duration in min.}>, <\text{activity type}>) \) schedules an activity with given activity type \textit{activityType} and duration \textit{duration}, i.e. adds it at the end of the list of activities between stops \( i \) and \( i + 1 \). The update of the driver status is done accordingly and the algorithm used can be seen as an extension to optional rules of the label-update made in the labeling algorithms of Goel (2009). The pseudo code is given in Algorithm 3.

**Algorithm 3** Update driver status

1: \( ptwr \leftarrow ptwr + \text{duration} \)
2:
3: \textbf{switch} \textit{activityType}
2.8. Myopic algorithm - A heuristic

4:    
5:      case drive
6:          udt ← udt + duration
7:          ddt ← ddt + duration
8:          ptr ← ptr + duration
9:          break
10:     end case
11:    
12:      case work
13:          ptr ← ptr + duration
14:          break
15:     end case
16:    
17:      case wait
18:          ptr ← ptr + duration
19:          break
20:     end case
21:    
22:      case break
23:          ptr ← ptr + duration
24:          if duration > 15 then
25:              udt ← 0
26:              hpb ← 0
27:          else
28:              hpb ← 1
29:          end if
30:          break
31:     end case
32:    
33:      case rest
34:          udt ← 0
35:          if duration = 180 then
36:              hpr ← 1
37:              ptr ← ptr + 180
38:          else
39:              ddt ← 0
40:              ptr ← 0
41:              hpb ← 0
42:              hpr ← 0
43:              red ← 0
44:              dte ← 0
45:          end if
46:          break
47:     end case
48:    
49:      case redrest
50:          udt ← 0
2.8.3. Determining the first reachable time window

Algorithm 4 is used to determine the first reachable time window. This is for example needed in Algorithm 5 if a driving time extension is considered, or in Algorithm 6 right at the beginning (see Section 2.8.4).

**Algorithm 4** Determine first reachable time window

**Input:** last stop $i$, potential arrival time $time$ at customer location $i + 1$, time window information for $i + 1$ (next stop)

**Output:** first time window reachable without incurring lateness ignoring a potential daily rest period still to take or last time window if lateness is not avoidable

1. $z \leftarrow 0$
2. $minLateness \leftarrow \max\left(time - TW_{end}^{i+1,0}, 0\right)$
3. for $k = 1$ to $noTW_i - 1$ do
4. \hspace{1em} $lateness \leftarrow \max\left(time - TW_{end}^{i+1,k}, 0\right)$
5. \hspace{1em} if \((lateness < minLateness) \lor (time \leq TW_{end}^{i+1,k} \land TW_{begin}^{i+1,k} < TW_{begin}^{i+1,z})\) then
6. \hspace{2em} $minLateness = lateness$
7. \hspace{2em} $z \leftarrow k$
8. \hspace{1em} end if
9. end for
10. return $z$

2.8.4. Scheduling activities for each pair of consecutive locations

The procedure in the step "Scheduling activities on arc $(i,i+1)$" depicted in Figure 2.21 is similar to the naive method for scheduling driving periods, breaks, and rest periods presented by Goel (2009) and is illustrated in detail in Figure 2.22. The corresponding
pseudo code is presented as Algorithm 5. In addition to Goel (2009), we also consider the possibility to extend the daily driving time by one hour.

Figure 2.22.: Scheduling activities on arc \((i, i + 1)\)
Algorithm 5 Scheduling activities on arc \((i,i+1)\)

1: // Schedule activities "between" stops \(i\) and \(i+1\).
2: // (Durations of daily rest periods may be modified later.)
3: //
4: //
5: 
6: \(duration \leftarrow 0\)
7: \(drivingTimeToDest \leftarrow \Delta_{drive}^{(i,i+1)}\)
8: 
9: while \(drivingTimeToDest > 0\) do
10:   // Determine the next driving time interval as the minimum of the nonstop
11:   // driving time left, the daily driving time left, the time until the next daily rest
12:   // period and the driving time still needed to reach the next stop.
13:   // If a partial daily rest period was made or it was decided previously that the
14:   // next daily rest period will be a reduced one, add two hours to the time until
15:   // the next daily rest period has to start.
16:   //
17:   \(duration \leftarrow \min \left(\begin{array}{c}
270 - udl,
540 + 60 \text{ dte} - \text{ ddt},
780 + 120 \text{ hpr} + 120 \text{ red} - \text{ ptr},
drivingTimeToDest
\end{array}\right)\)
18: 
19:   scheduleActivity \(i, duration, drive\)
20: 
21: Update driver status
22: 
23: \(drivingTimeToDest \leftarrow drivingTimeToDest - duration\)
24: 
25: if \(drivingTimeToDest = 0\) then
26:   break
27: 
28: else
29: 
30: // If less than one hour of driving is left to reach the next stop, take a driving
31: // time extension if possible and advantageous.
32: //
33: // A driving time extension is considered
34: // - if at most one hour of driving is left until the next stop is reached,
35: // - if at least one daily rest period on the current arc has already been made,
36: // - if in the current week, less than two driving time extensions have been taken
37: // - if it is possible to save a daily rest period on the current arc and thus
38: // - reach an earlier time window or reduce lateness.
2.8. Myopic algorithm - A heuristic

\[
\begin{align*}
&\text{if } \left( \begin{array}{c}
\text{drivingTimeToDest} \leq 60 \\
\land \text{getDailyRestP osSize() > 0} \\
\land \text{ddt} = 540 \\
\land \text{noExt < 2}
\end{array} \right) \text{ then} \\
&\quad \text{redRestPoss} \leftarrow 0
\end{align*}
\]

\[
\begin{align*}
&\text{if } \text{noRed < 3} \text{ then} \\
&\quad \text{redRestPoss} \leftarrow 1
\end{align*}
\]

\[
\begin{align*}
&\text{if } \left( \begin{array}{c}
780 + 120 \text{ redRestPoss} \\
\geq \text{ptr} + \text{drivingTimeToDest} + 45 + \text{workingTime}
\end{array} \right) \text{ then} \\
&\quad \text{time1} \leftarrow \text{ptwr} + 45 + \text{drivingTimeToDest} \\
&\quad \text{time2} \leftarrow \text{ptwr} + 660 + \text{drivingTimeToDest}
\end{align*}
\]

// Determine the first reachable time window for both alternatives
// (Algorithm 4).

\[
\begin{align*}
&z_1 \leftarrow \text{Determine first reachable time window for time1} \\
&z_2 \leftarrow \text{Determine first reachable time window for time2}
\end{align*}
\]

// If the first reachable time window starts earlier or the lateness is less if
// a driving time extension is used, take the driving time extension.

\[
\begin{align*}
&\text{if } \text{TW}^{\text{begin}}_{i+1,z1} < \text{TW}^{\text{begin}}_{i+1,z2} \lor \text{time2} - \text{TW}^{\text{end}}_{i+1,z2} > 0 \text{ then} \\
&\quad \text{scheduleActivity}(i, 45, \text{break}) \\
&\quad \text{Update driver status}
\end{align*}
\]

\[
\begin{align*}
&\text{scheduleActivity}(i, \text{drivingTimeToDest, drive}) \\
&\quad \text{Update driver status}
\end{align*}
\]

\[
\begin{align*}
&dte \leftarrow 1 \\
&\text{noExt} \leftarrow \text{noExt} + 1
\end{align*}
\]

\[
\begin{align*}
&\text{if } 780 < \text{ptr} + \bar{\Delta}_{i+1}^{\text{service}} \text{ then} \\
&\quad \text{red} = 1
\end{align*}
\]

// If the daily driving time or the time until the next daily rest period is
// exhausted, take a daily rest period. Otherwise, the nonstop driving time
// equals 4.5 hours and a break has to be taken.
if \( ddt = 780 \lor ptr + 45 - 15 \ hpb \geq 780 \) then

\[
duration \leftarrow 660 - 120 \ (hpr + red)
\]

if \( red = 1 \) then

\[
scheduleActivity(i, duration, redrest)
\]

else

\[
scheduleActivity(i, duration, rest)
\]

end if

Update driver status

else

\[
duration \leftarrow 45 - 15 \ hpb
\]

\[
scheduleActivity(i, duration, rest)
\]

Update driver status

end if

\[
\text{if } \text{drivingTimeToDest} = 0
\]

end while

In the step "Choose time window at stop \( i + 1 \)", the time window at the next stop \( i + 1 \) is chosen. Therefore, the first reachable time window, that means the first time window that ends after the last activity scheduled so far, is determined\(^{44}\). When considering the current schedule, a daily rest period may be necessary as the time left does not suffice to wait, load and/or unload goods at customer location \( i + 1 \) because of the maximum time interval between two daily rest periods (standard rule 6). If scheduling a daily rest period would lead to lateness, it is tried to reduce the durations of daily rest periods scheduled on the arc \((i, i + 1)\). If this does not help to be on time, it is tested if it is possible to extend the duration of the previous daily rest period and thus eliminate waiting time. In this way, the start of the time interval between the last daily rest period and the next one that is not scheduled yet can be postponed. The option to plan the next daily rest period to be a reduced one is also taken into account.\(^{45}\) If with this modification the daily rest period can take place after loading and/or unloading goods at the customer, the schedule is altered accordingly. The method \textit{extendRestDurationLastRest (< modifier in min. >)} in Algorithm 6 extends the duration of the last scheduled daily rest period and modifies \( ptwr \) and the starting times of subsequent activities. In case the above modifications do not suffice to reach the time window, the next time window\(^{46}\) is selected. These steps are repeated if either it is possible to be on time or the last time window is selected. The flowchart in Figure 2.23 illustrates the course of action. The pseudo code is given in Algorithm 6.

\(^{44}\) The corresponding pseudo code is given in Algorithm 4 in Section 2.8.3

\(^{45}\) If the next daily rest period is planned to be a reduced one, the maximum time between the last daily rest period and the following one increases by 2 hours.

\(^{46}\) Time windows have to be sorted by their start time and shall not overlap.
Figure 2.23.: Choose time window at stop $i + 1$
Algorithm 6 Choose time window at stop $i + 1$

1: //
2: // Determine the first reachable time window. Plan an additional daily rest period
3: // if it is necessary before loading or unloading may start. If this causes lateness
4: // for the time window currently considered, first try to reduce the duration of daily
5: // rest periods on this arc. If this does not work, try to leave out the last daily
6: // rest period:
7: // If waiting time occurs, compensate it if possible, by extending the previous daily
8: // rest period on this arc. Additionally, consider the option to plan the next daily
9: // rest period to be a reduced one to obtain two additional hours until the next daily
10: // rest period is necessary.
11: //
12: $z \leftarrow$ Determine first reachable time window for $ptwr$
13:  
14: repeat
15:     $chosenTWEnd \leftarrow TW_{i+1,z}^{end}$
16:     $z \leftarrow z + 1$
17:     $waitingTime \leftarrow \max\left(0, TW_{i+1,z}^{begin} - ptwr\right)$
18:     $dailyTimeAfterService \leftarrow ptr + waitingTime + \Delta_{i+1}^{service}$
19:     if $dailyTimeAfterService > 780 + 120 (hpr + red)$ then
20:         // Without daily rest period, the time does not suffice to wait and serve the
21:         // customer. Try to schedule a daily rest period.
22:          
23: if $(hpr = 1) \lor (red = 1)$ then
24:     duration $\leftarrow 540$
25: else
26:     duration $\leftarrow 660$
27: end if
28: if $ptwr + duration > chosenTWEnd$ then
29:     // If lateness occurs, test, whether reducing rest periods on the current arc
30:     // helps to reach the chosen time window in time.
31:     //
32:     // Determine the number of daily rest periods on the current arc that may be
33:     // reduced.
34:     $posNoReductions \leftarrow getNoReducableRestPeriods()$
2.8. Myopic algorithm - A heuristic

if \( hpr = 0 \) \( \land \) \( red = 0 \) then
\( posNoReductions \leftarrow posNoReductions + 1 \)
end if

\( posNoReductions = \min (posNoReductions, 3 - noRed) \)

if \( ptwr + duration - 120 \ posNoReductions \leq chosenTW \) then
\(\text{if } red = 0 \text{ then}\)
\(\text{scheduleActivity}(i, duration, rest)\)
\(\text{else}\)
\(\text{scheduleActivity}(i, duration, redrest)\)
end if

Update driver status

break

else

// If it is not possible to schedule a daily rest period without lateness, try
// to extend the last daily rest period on this arc by the waiting time
// to shift the 24 h hours time interval and/or try to extend it by deciding
// that the next daily rest period should be a reduced one.

\( reducedRestPoss = 0 \)

if \( noRed < 3 \) \( \land \) \( hpr = 0 \) then
\( reducedRestPoss = 1 \)
end if

\( hadDailyRest = 0 \)

if \( \text{getNoReducableRestPeriods()} > 0 \) then
\( hadDailyRest = 1 \)
end if

\(\left( (dailyTimeAfterService - \text{hadDailyRest} \cdot \text{waitingTime} \leq 780 + 120 (hpr + reducedRestPoss) \land \text{getNoRestPeriods()} > 1 \right) \) then
\(\text{if } hadDailyRest > 0 \text{ then}\)
\(\text{extendRestDurationLastRest}(\text{waitingTime})\)
end if
// if hadDailyRest > 0

\(\text{if } ptr + \text{waitingTime} + \Delta^{serv}_{i+1} > 780 \land red = 0 \text{ then}\)
\( red \leftarrow 1 \)
end if
2. Scheduling of driving times, breaks and rest periods

break

else

// A daily rest period is necessary, but it is not possible to avoid
// lateness if this time window is chosen. Schedule the daily rest
// period.

if red = 0 then
    scheduleActivity(i, duration, rest)
else
    scheduleActivity(i, duration, redrest)
end if

Update driver status

end if

end if

Update driver status

break

end if

else

// The time suffices to take a regular daily rest period. Note that the daily
// rest period may end after the start of the time window.

scheduleActivity(i, duration, rest)

Update driver status

break

end if

end if

else

// An additional daily rest period is not necessary.

// If lateness occurs, the current time window considered is already the last
// one, as we started this loop with the "first reachable time window".

break

end if

end if

lateness ← max(0, ptwr - chosenTWEnd)

until lateness = 0 ∨ z = noTW_{i+1}

// In the repeat-loop, z was raised by 1 one time too often. Therefore, subtract 1.

z ← z - 1
In the step "Modify rest durations and schedule activities at stop $i + 1$" (Figure 2.24 and Algorithm 7), if there is lateness, regular daily rest periods are reduced if possible, and activities are rescheduled accordingly. Afterwards, activities at stop $i + 1$ are scheduled. If there is still time left until the start of the time window chosen, potential waiting time can be compensated by a resting activity. The options to take a partial daily rest period or a partial break are included. If there is still waiting time, we try to compensate it by extending the last daily rest period on the current arc. Finally, if the stop is a customer location, loading and/or unloading is scheduled.

The method $reduceRestDurationLastRest()$ in Algorithm 7 is used to decrease the the duration of the last unreduced daily rest period on the considered arc if this helps to decrease lateness. Waiting time may be compensated by extending the last daily rest period on the current arc (method $extendRestDurationLastRest(< extension in min. >)$).

Note that the methods
- $getNoReducableRestPeriods()$,  
- $reduceRestDurationLastRest()$ and  
- $extendRestDurationLastRest(< extension in min. >)$

refer to two lists, the list with all daily rest period positions and the list with regular daily rest period positions (that may still be reduced), that are reset after each iteration of the algorithm.
2. Scheduling of driving times, breaks and rest periods

Figure 2.24.: Modify rest durations and schedule activities at stop i + 1

- Time window has been chosen at stop i + 1
- Lateness: no
- Reduce the duration of daily rest periods on arc (i, i+1) until:
  - no additional reduction is possible or
  - lateness is avoided
- Waiting time: no
- Resting activity at stop i + 1 to compensate waiting time. If the time suffices, take:
  - a (reduced) second part of a daily rest period or
  - a first part of a daily rest period or
  - a break or
  - a first part of a break
- Waiting time: yes
- Daily rest period on arc (i, i+1): no
- Extend previous daily rest period to compensate waiting time
- Customer location: no
- Load / unload goods
- Wait
- Yes
- Customer location: no
- Load / unload goods
- Wait
- Yes
Algorithm 7 Modify rest durations and schedule activities at stop $i + 1$

1: $lateness \leftarrow \max\left(0, ptwr - TW_{i+1,z}^{end}\right)$
2: 
3: // If there is lateness, reduce the duration of daily rest periods.
4: 
5: if $lateness > 0$ then
6: 
7: $posNoReductions = \min(getNoReducableRestPeriods(), 3 - noRed)$
8: 
9: while ($ptwr > TW_{i+1,z}^{end}$) $\land$ ($posNoReductions > 0$) do
10: 
11: reduceRestDurationLastRest()
12: 
13: $posNoReductions \leftarrow posNoReductions - 1$
14: 
15: end while
16: 
17: else
18: 
19: // Plan activities after the arrival at the customer location.
20: 
21: // If there is waiting time, try to compensate it by resting activities.
22: 
23: // If the last activity scheduled was a (reduced) daily rest period ignore this step.
24: 
25: $noActivities \leftarrow getNoActivitiesArc()$
26: 
27: if $noActivities < 1 \lor getActivityType(noActivities - 1) \neq "\text{rest}"$ then
28: 
29: if $hpr = 1 \lor red = 1$ then
30: 
31: $duration \leftarrow 540$
32: 
33: else
34: 
35: $duration \leftarrow 660$
36: 
37: end if
38: 
39: if $ptwr \leq TW_{i+1,z}^{begin} - duration$ then
40: 
41: if $red = 1$ then
42: 
43: scheduleActivity $(i, duration, redrest)$
44: 
45: Update driver status
46: 
47: else
48: 
49: scheduleActivity $(i, duration, rest)$
50: 
51: Update driver status
52: 
53: end if
54: 
55: else if ($ptwr \leq TW_{i+1,z}^{begin} - 540) \land (noRed < 3)$ then
56: 
57: scheduleActivity $(i, 540, redrest)$
58: 
59: Update driver status
60: 
61: end if
2. Scheduling of driving times, breaks and rest periods

else if \((hpr = 0) \land (red = 0) \land (ptwr \leq TW_{i+1,z} - 180)\) then

\[\text{scheduleActivity}(i, 180, \text{rest})\]

Update driver status

else if \((ptwr \leq TW_{i+1,z} - 45 + 15 \times hpb) \land (udt > 0)\) then

\[\text{scheduleActivity}(i, 45 - 15 \times hpb, \text{break})\]

Update driver status

else if \((ptwr \leq TW_{i+1,z} - 15) \land (hpb = 0)\) then

\[\text{scheduleActivity}(i, 15, \text{rest})\]

Update driver status

end if  // if \(ptwr \leq TW_{i+1,z} - \text{duration}\)

end if  // if noActivities < 1 \lor \text{getActivityType}(\text{noActivities} - 1) \neq "rest"

end if  // if lateness > 0

// Postprocessing: Compensate waiting time by extending the duration of the last
daily rest period if possible.

if \(ptwr \leq TW_{i+1,z}\) then

if \(\text{getDailyRestPosSize()} > 0\) then

\[\text{extendRestDurationLastRest}(TW_{i+1,z} - ptwr)\]

else

Save the driver status.

Save the chosen time window.

\[\text{scheduleActivity}(i, TW_{i+1,z} - ptwr, \text{wait})\]

end if  // if \(\text{getDailyRestPosSize()} > 0\)

else

Save the driver status.

Save the chosen time window.

end if  // if \(ptwr \leq TW_{i+1,z}\)

// Plan loading or unloading at customer location.

if \(\Delta_{i+1} > 0\) then

\[\text{scheduleActivity}(i, \Delta_{i+1}, \text{work})\]

end if  // if \(\Delta_{i+1} > 0\)
2.9. Numerical experiments - Part 2

In this section, we present the results obtained with the myopic heuristic and compare them to the results achieved with the MILP models. An example of schedules that were obtained with the different planning techniques for one of the instances considered in the numerical experiments is given in Section 2.9.1. In Section 2.9.2 lateness and overall schedule duration are compared generally. Some managerial insights are discussed in Section 2.9.3.

2.9.1. Example of schedules

We illustrate how the different planning techniques behave by means of an example. Base instance 3 (see Figure 2.25) with 3 alternative time windows and a time window size of 30 minutes (see Table 2.6) was chosen for the analysis.

![Figure 2.25: Route of base instance 3](image)
Table 2.6.: Time windows

Table 2.7 (2.9) illustrates the detailed schedule obtained by using the MILP model without (with) optional rules. Table 2.8 depicts the driver schedule obtained with the myopic heuristic.

It is interesting to see the advantages when optimizing over all arcs. Independent of the planning technique, the driver starts his work week at the first customer (no driving duration between start vertex 0 and customer vertex 1) at 7:47 on Monday morning. While the models decide to take the first time window and thus accept a lateness of 47 minutes, the myopic algorithm decides to avoid lateness at the first customer and chooses the second time window compensating part of the waiting time by a first partial break. The effect can be seen when having a look at the lateness at the second customer. While the myopic algorithm cannot avoid a lateness of 2:16 hours, lateness amounts to 1:03 hours when the models are used. Over the first two customers, lateness can thus be reduced by 26 minutes.

The choice of time windows varies between the three schedules. The use of reduced daily rest periods allows the driver to be on time at stop three, no matter if the model with optional rules or the myopic algorithm is chosen. On the last arc, the MILP model with consideration of the optional rules has the advantage to "know" that there are no remaining requests in the considered week. All possible driving time extensions and reduced daily rest periods are used such that the driver is able to finish his tour significantly earlier than in the schedule that was created by the myopic algorithm. In turn, the myopic algorithm achieves a much earlier completion time than the MILP model without consideration of the optional rules.
<table>
<thead>
<tr>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>(un)load</td>
<td>07:47</td>
<td>09:47</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**stop: 1**

chosen time window:

start:  
Mon 06:30

end:  
Mon 07:00

**calculated arrival:**  
Mon 07:47

**lateness:**  
00:47

`stop: 2`

chosen time window:

start:  
Mon 10:30

end:  
Mon 11:00

**calculated arrival:**  
Mon 12:03

**lateness:**  
01:03

<table>
<thead>
<tr>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>drive</td>
<td>14:03</td>
<td>16:17</td>
<td>02:14</td>
</tr>
<tr>
<td>Mon</td>
<td>break</td>
<td>16:17</td>
<td>17:02</td>
<td>00:45</td>
</tr>
<tr>
<td>Mon</td>
<td>drive</td>
<td>17:02</td>
<td>20:47</td>
<td>03:45</td>
</tr>
<tr>
<td>Mon</td>
<td>rest</td>
<td>20:47</td>
<td>07:47</td>
<td>11:00</td>
</tr>
<tr>
<td>Tue</td>
<td>drive</td>
<td>07:47</td>
<td>12:17</td>
<td>04:30</td>
</tr>
<tr>
<td>Tue</td>
<td>break</td>
<td>12:17</td>
<td>13:02</td>
<td>00:45</td>
</tr>
<tr>
<td>Tue</td>
<td>drive</td>
<td>13:02</td>
<td>17:32</td>
<td>04:30</td>
</tr>
<tr>
<td>Tue</td>
<td>rest</td>
<td>17:32</td>
<td>04:32</td>
<td>11:00</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>04:32</td>
<td>09:02</td>
<td>04:30</td>
</tr>
<tr>
<td>Wed</td>
<td>break</td>
<td>09:02</td>
<td>09:47</td>
<td>00:45</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>09:47</td>
<td>12:07</td>
<td>02:20</td>
</tr>
<tr>
<td>Wed</td>
<td>(un)load</td>
<td>12:07</td>
<td>14:07</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**stop: 3**

chosen time window:

start:  
Wed 10:30

end:  
Wed 11:00

**calculated arrival:**  
Wed 12:07

**lateness:**  
01:07

<table>
<thead>
<tr>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed</td>
<td>drive</td>
<td>14:07</td>
<td>16:17</td>
<td>02:10</td>
</tr>
<tr>
<td>Wed</td>
<td>rest</td>
<td>16:17</td>
<td>04:55</td>
<td>12:38</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>04:55</td>
<td>06:00</td>
<td>01:05</td>
</tr>
<tr>
<td>Thu</td>
<td>(un)load</td>
<td>06:00</td>
<td>08:00</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**stop: 4**

chosen time window:

start:  
Thu 06:00

end:  
Thu 06:30

**calculated arrival:**  
Thu 06:00

**lateness:**  
00:00

<table>
<thead>
<tr>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thu</td>
<td>drive</td>
<td>08:00</td>
<td>11:25</td>
<td>03:25</td>
</tr>
<tr>
<td>Thu</td>
<td>break</td>
<td>11:25</td>
<td>12:10</td>
<td>00:45</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>12:10</td>
<td>16:40</td>
<td>04:30</td>
</tr>
<tr>
<td>Thu</td>
<td>rest</td>
<td>16:40</td>
<td>03:40</td>
<td>11:00</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>03:40</td>
<td>08:10</td>
<td>04:30</td>
</tr>
<tr>
<td>Fri</td>
<td>break</td>
<td>08:10</td>
<td>08:55</td>
<td>00:45</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>08:55</td>
<td>13:25</td>
<td>04:30</td>
</tr>
<tr>
<td>Fri</td>
<td>rest</td>
<td>13:25</td>
<td>00:25</td>
<td>11:00</td>
</tr>
<tr>
<td>Sat</td>
<td>drive</td>
<td>00:25</td>
<td>03:07</td>
<td>02:42</td>
</tr>
</tbody>
</table>

**stop: 5**

chosen time window:

start:  
Mon 00:00

end:  
Sun 23:59

**calculated arrival:**  
Sat 03:07

**lateness:**  
00:00

Table 2.7.: Optimal schedule identified by the MILP model without optional rules
<table>
<thead>
<tr>
<th>day</th>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>Mon</td>
<td>break</td>
<td>07:47</td>
<td>08:02</td>
<td>00:15</td>
</tr>
<tr>
<td>Mon</td>
<td>Mon</td>
<td>wait</td>
<td>08:02</td>
<td>09:00</td>
<td>00:58</td>
</tr>
<tr>
<td>Mon</td>
<td>Mon</td>
<td>(unload)</td>
<td>09:00</td>
<td>11:00</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**stop: 1**

**chosen time window:**

**start:**

<table>
<thead>
<tr>
<th>day</th>
<th>from</th>
<th>until</th>
<th>Mon 09:00</th>
<th>end: Mon 09:30</th>
</tr>
</thead>
</table>

**calculated arrival:**

**lateness:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>11:00</th>
<th>13:16</th>
<th>02:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>(unload)</td>
<td></td>
<td>13:16</td>
<td>15:16</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**stop: 2**

**chosen time window:**

**start:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>15:16</th>
<th>17:30</th>
<th>02:14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>break</td>
<td>17:30</td>
<td>18:00</td>
<td>00:30</td>
</tr>
<tr>
<td>Mon</td>
<td>drive</td>
<td>18:00</td>
<td>20:47</td>
<td>02:47</td>
</tr>
<tr>
<td>Mon</td>
<td>rest</td>
<td>20:47</td>
<td>05:47</td>
<td>09:00</td>
</tr>
<tr>
<td>Tue</td>
<td>drive</td>
<td>05:47</td>
<td>10:17</td>
<td>04:30</td>
</tr>
<tr>
<td>Tue</td>
<td>break</td>
<td>10:17</td>
<td>11:02</td>
<td>00:45</td>
</tr>
<tr>
<td>Tue</td>
<td>drive</td>
<td>11:02</td>
<td>15:32</td>
<td>04:30</td>
</tr>
<tr>
<td>Tue</td>
<td>rest</td>
<td>15:32</td>
<td>01:57</td>
<td>10:25</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>01:57</td>
<td>06:27</td>
<td>04:30</td>
</tr>
<tr>
<td>Wed</td>
<td>break</td>
<td>06:27</td>
<td>07:12</td>
<td>00:45</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>07:12</td>
<td>10:30</td>
<td>03:18</td>
</tr>
<tr>
<td>Wed</td>
<td>(unload)</td>
<td>10:30</td>
<td>12:30</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**calculated arrival:**

**lateness:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>12:30</th>
<th>13:42</th>
<th>01:12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed</td>
<td>rest</td>
<td>13:42</td>
<td>00:42</td>
<td>11:00</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>00:42</td>
<td>02:45</td>
<td>02:03</td>
</tr>
<tr>
<td>Thu</td>
<td>break</td>
<td>02:45</td>
<td>03:30</td>
<td>00:45</td>
</tr>
<tr>
<td>Thu</td>
<td>(unload)</td>
<td>03:30</td>
<td>05:30</td>
<td>02:00</td>
</tr>
</tbody>
</table>

**stop: 3**

**chosen time window:**

**start:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>03:30</th>
<th>end: Wed 11:00</th>
</tr>
</thead>
</table>

**calculated arrival:**

**lateness:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>05:30</th>
<th>10:00</th>
<th>04:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thu</td>
<td>break</td>
<td>10:00</td>
<td>10:45</td>
<td>00:45</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>10:45</td>
<td>13:12</td>
<td>02:27</td>
</tr>
<tr>
<td>Thu</td>
<td>rest</td>
<td>13:12</td>
<td>00:12</td>
<td>11:00</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>00:12</td>
<td>04:42</td>
<td>04:30</td>
</tr>
<tr>
<td>Fri</td>
<td>break</td>
<td>04:42</td>
<td>05:27</td>
<td>00:45</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>05:27</td>
<td>09:57</td>
<td>04:30</td>
</tr>
<tr>
<td>Fri</td>
<td>rest</td>
<td>09:57</td>
<td>20:57</td>
<td>11:00</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>20:57</td>
<td>00:37</td>
<td>03:40</td>
</tr>
</tbody>
</table>

**stop: 4**

**chosen time window:**

**start:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>03:30</th>
<th>end: Thu 04:00</th>
</tr>
</thead>
</table>

**calculated arrival:**

**lateness:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>05:30</th>
<th>10:00</th>
<th>04:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thu</td>
<td>break</td>
<td>10:00</td>
<td>10:45</td>
<td>00:45</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>10:45</td>
<td>13:12</td>
<td>02:27</td>
</tr>
<tr>
<td>Thu</td>
<td>rest</td>
<td>13:12</td>
<td>00:12</td>
<td>11:00</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>00:12</td>
<td>04:42</td>
<td>04:30</td>
</tr>
<tr>
<td>Fri</td>
<td>break</td>
<td>04:42</td>
<td>05:27</td>
<td>00:45</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>05:27</td>
<td>09:57</td>
<td>04:30</td>
</tr>
<tr>
<td>Fri</td>
<td>rest</td>
<td>09:57</td>
<td>20:57</td>
<td>11:00</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>20:57</td>
<td>00:37</td>
<td>03:40</td>
</tr>
</tbody>
</table>

**stop: 5**

**chosen time window:**

**start:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>00:00</th>
<th>end: Sun 23:59</th>
</tr>
</thead>
</table>

**calculated arrival:**

**lateness:**

<table>
<thead>
<tr>
<th>day</th>
<th>drive</th>
<th>00:00</th>
<th>20:57</th>
<th>23:59</th>
</tr>
</thead>
</table>

Table 2.8.: Schedule created with the myopic heuristic
<table>
<thead>
<tr>
<th>date</th>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop: 1</td>
<td>Mon</td>
<td>(un)load</td>
<td>07:47</td>
<td>09:47</td>
<td>02:00</td>
</tr>
<tr>
<td>chosen time window:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start:</td>
<td>Mon</td>
<td></td>
<td>06:30</td>
<td></td>
<td>Mon 07:00</td>
</tr>
<tr>
<td>calculated arrival:</td>
<td></td>
<td></td>
<td>07:47</td>
<td></td>
<td>00:47</td>
</tr>
<tr>
<td>lateness:</td>
<td>Mon</td>
<td>drive</td>
<td>09:47</td>
<td></td>
<td>12:03 02:16</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>(un)load</td>
<td>12:03</td>
<td></td>
<td>14:03 02:00</td>
</tr>
<tr>
<td>stop: 2</td>
<td>Mon</td>
<td></td>
<td>10:30</td>
<td></td>
<td>Mon 11:00</td>
</tr>
<tr>
<td>chosen time window:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start:</td>
<td>Mon</td>
<td></td>
<td>07:47</td>
<td></td>
<td>Mon 08:00</td>
</tr>
<tr>
<td>calculated arrival:</td>
<td></td>
<td></td>
<td>12:03</td>
<td></td>
<td>Mon 01:03</td>
</tr>
<tr>
<td>lateness:</td>
<td>Mon</td>
<td>drive</td>
<td>14:03</td>
<td></td>
<td>16:17 02:14</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>break</td>
<td>16:17</td>
<td></td>
<td>17:02 00:45</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>drive</td>
<td>17:02</td>
<td></td>
<td>21:52 04:30</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>rest</td>
<td>21:32</td>
<td></td>
<td>06:32 09:00</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>drive</td>
<td>06:32</td>
<td></td>
<td>11:02 04:30</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>break</td>
<td>11:02</td>
<td></td>
<td>11:47 00:45</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>drive</td>
<td>11:47</td>
<td></td>
<td>16:17 04:30</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>rest</td>
<td>16:17</td>
<td></td>
<td>01:17 09:00</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>drive</td>
<td>01:17</td>
<td></td>
<td>05:47 04:30</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>break</td>
<td>05:47</td>
<td></td>
<td>06:32 00:45</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>drive</td>
<td>06:32</td>
<td></td>
<td>08:07 01:35</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>(un)load</td>
<td>08:07</td>
<td></td>
<td>10:07 02:00</td>
</tr>
<tr>
<td>stop: 3</td>
<td>Wed</td>
<td></td>
<td>08:00</td>
<td></td>
<td>Wed 08:30</td>
</tr>
<tr>
<td>chosen time window:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start:</td>
<td>Wed</td>
<td></td>
<td>08:00</td>
<td></td>
<td>Wed 00:00</td>
</tr>
<tr>
<td>calculated arrival:</td>
<td></td>
<td></td>
<td>10:07</td>
<td></td>
<td>Wed 13:02 02:55</td>
</tr>
<tr>
<td>lateness:</td>
<td>Wed</td>
<td>drive</td>
<td>13:02</td>
<td></td>
<td>13:53 03:53</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>rest</td>
<td>13:02</td>
<td></td>
<td>01:55 03:20</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>drive</td>
<td>02:55</td>
<td></td>
<td>03:15 00:15</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>break</td>
<td>03:15</td>
<td></td>
<td>03:30 00:15</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>(un)load</td>
<td>03:30</td>
<td></td>
<td>05:30 02:00</td>
</tr>
<tr>
<td>stop: 4</td>
<td>Thu</td>
<td></td>
<td>03:30</td>
<td></td>
<td>Thu 04:00</td>
</tr>
<tr>
<td>chosen time window:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start:</td>
<td>Thu</td>
<td></td>
<td>03:30</td>
<td></td>
<td>Thu 04:00</td>
</tr>
<tr>
<td>calculated arrival:</td>
<td></td>
<td></td>
<td>05:30</td>
<td></td>
<td>Thu 09:40 04:10</td>
</tr>
<tr>
<td>lateness:</td>
<td>Thu</td>
<td>drive</td>
<td>05:30</td>
<td></td>
<td>09:40 00:30</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>break</td>
<td>09:40</td>
<td></td>
<td>10:10 00:30</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>drive</td>
<td>10:10</td>
<td></td>
<td>14:40 04:30</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>break</td>
<td>14:40</td>
<td></td>
<td>15:25 00:45</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>drive</td>
<td>15:25</td>
<td></td>
<td>16:25 01:00</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>rest</td>
<td>16:25</td>
<td></td>
<td>01:25 09:00</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>01:25</td>
<td></td>
<td>05:55 04:30</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>break</td>
<td>05:55</td>
<td></td>
<td>06:40 00:45</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>06:40</td>
<td></td>
<td>11:10 04:30</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>break</td>
<td>11:10</td>
<td></td>
<td>11:55 00:45</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>11:55</td>
<td></td>
<td>12:52 00:57</td>
</tr>
<tr>
<td>stop: 5</td>
<td>Mon</td>
<td></td>
<td>00:00</td>
<td></td>
<td>Sun 23:59</td>
</tr>
<tr>
<td>chosen time window:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start:</td>
<td>Mon</td>
<td></td>
<td>00:00</td>
<td></td>
<td>Mon 12:52</td>
</tr>
<tr>
<td>calculated arrival:</td>
<td></td>
<td></td>
<td>12:52</td>
<td></td>
<td>Mon 00:00</td>
</tr>
<tr>
<td>lateness:</td>
<td>Fri</td>
<td>drive</td>
<td>12:52</td>
<td></td>
<td>Mon 00:00</td>
</tr>
</tbody>
</table>

Table 2.9.: Optimal schedule identified by the MILP model with optional rules
2.9.2. Comparison of the myopic heuristic and the MILP models

The run time of the myopic heuristic was less than 1 millisecond for each test instance. In Figures 2.26 and 2.27, lateness and overall schedule duration of the schedules constructed by the myopic heuristic and those constructed with the MILP models (with and without optional rules) are compared.

![Graph showing comparison of lateness (min.) across various base instances.](image)

**Figure 2.26:** Average lateness of schedules depending on the solution technique

Figure 2.26 shows that, on average, the myopic algorithm performs better than the model without optional rules. Lateness decreases by 18% when the myopic algorithm is used. For all instances with base instances 1, 2, 4, 6, 7, and 9, there is no difference concerning lateness no matter if the model with optional rules or the myopic heuristic is chosen as solution technique. But the cumulated lateness over all instances is 83% worse if the heuristic instead of the MILP model with the consideration of optional rules is used to determine the driver schedule. This shows that the MILP model with consideration of the optional rules has a significant higher potential for keeping lateness low or avoiding it. The reason for this is that for the choice of time windows and the determination of driver activities the whole tour is considered when using the MILP models. In contrast, the myopic heuristic plans driver activities successively, not considering stops beyond the next one.

The overall schedule duration (see Figure 2.27) is on average slightly reduced (less than 1%) if the myopic heuristic is chosen and not the model without optional rules, but compared to the model with optional rules, the schedules constructed with the myopic heuristic require on average 5% more time. This is significant as discussed in Section 2.7.3.
2.9.3. Managerial insights

The following figures depict schedule properties retrieved with each of the three solution techniques described in the previous sections, depending on input parameters and settings.

At first, we analyze the influence of the time window properties on the overall lateness. As one might have expected, lateness tends to be less if more time windows are available. When generating test instances from base instances, the second time window added for a chosen base instance is set to start after a time interval that follows the first time window. In the driver schedules obtained for those instances, lateness is significantly less compared with the corresponding schedules for the instances with only one time window.\(^{47}\) When the third time window at a customer location is added, which ends before the start of the first time window assigned, still a reduction can be noticed in all but one test set (see Figure 2.28). As the myopic heuristic only identifies local optima, deterioration is possible when time windows are added. Neglecting the possibility to use waiting time to start a daily rest period or break, and choosing a time window earlier even if this causes lateness can lead to more lateness at subsequent customer locations. As expected, lateness is reduced if time windows are extended leaving their starting time constant. If two or more time windows are available with 10 hours length (simulation of opening hours), no lateness is observed at all no matter which solution technique is chosen.

\(^{47}\) For the generation of test instances from base instances and the assignment of time windows see Section 2.7.1.
The schedule duration does not behave as uniformly as the lateness (see Figure 2.29), since it is only considered as a subordinate optimization criterion in the MILP models.
While the average schedule duration decreases with the number and the length of time windows if the model with optional rules is chosen, with the model without optional rules, the average schedule duration sometimes increases when the second time window is added. With the myopic heuristic, all average values for two time windows show this deterioration.

Figure 2.30 shows the average proportions of the different driver activities. It is interesting that the proportion of working time (i.e. the driving time and the time needed for loading/unloading goods) with 45.81% considering the model with optional rules is reduced by about 6% if the optional rules are neglected and by nearly 5% if the myopic heuristic is chosen as solution technique. The proportions of rest periods, breaks and waiting time increase accordingly.

Research has shown that long periods of night work can be harmful for the health of workers and driving at night raises safety risks for the driver himself and other participants in traffic. Therefore, Directive 2002/15/EC (European Parliament and Council of the European Union (2002)) lays down basic rules for night work that have to be implemented in national laws. The rules incorporate the necessity to compensate night workers in accordance with national legislative measures. German law, i.e. the "Arbeitszeitgesetz" (Deutscher Bundestag (1994)), specifies that compensation may be a corresponding number of paid days off or an adequate surcharge on the gross remuneration. This means that the transport undertakings themselves may be interested in keeping drivers’ working hours at night as low as possible.

Neither the models nor the myopic algorithm incorporate the consideration of nightly
working time. Figure 2.31 shows the average proportions of nightly working time with no efforts taken to keep them low. For the night time we use the definition given by the "Arbeitszeitgesetz". There, the night time is defined to be the time period between 22:00 pm and 06:00 am. It is interesting to note that the average proportions of nightly working time for all solution techniques are less than the proportion of night time on the overall day with 24 hours (lower right corner of Figure 2.31). The reason for this may be the originally planned arrival times that served as basis for the definition of time windows. Only base instances 6, 8, 10, 13 and 14 contained arrival times which were planned during night time. Still, independent of the solution technique, the resulting working time at night is not negligible.

![Figure 2.31: Proportions of working time at night and day](image-url)
3. The sequence vehicle refueling problem with time windows

In the previous chapter, we focused on the scheduling of driver activities in accordance to Regulation (EC) No 561/2006. In this chapter, we will deal with the vehicle refueling subproblem.

Fuel represents an important cost driver in transportation logistics. Given a scheduled route, different factors influence the total associated fuel expenditure. Usually, the longer the total length of the route, the higher the fuel consumption will be. Minimizing the travel distance is therefore a typical planning objective in vehicle routing optimization.

However, fuel expenditure is not only impacted by the amount of fuel consumed by the vehicle but also by the fuel price itself. Fuel prices may vary significantly at different gas stations and therefore may have a strong impact on the total cost of vehicle routes. Naturally, transportation companies cannot influence the prices but they can make a selection among the gas stations. It is therefore reasonable to consider an approach that optimizes the cost of a schedule by including the choice of the refueling stops and the quantities of fuel to be purchased.

3.1. Problem Description

Again, a fixed sequence of customer locations with time windows is considered for a single vehicle. This time, rest periods and breaks of the driver are neglected. Instead, a choice among possible gas stations has to be made and optimal refueling quantities need to be determined. In the sequence vehicle refueling problem with time windows, a sequence of customer locations and the main route that has to be traveled by the vehicle to visit them is given. Geographical positions of gas stations along the route and the corresponding diesel prices are known. Driving durations and fuel consumptions between consecutive customer locations are additional input parameters as well as the starting time. For each customer location there is a single time window which defines a lower and an upper bound for the start of loading and/or unloading goods. The time that is needed for loading, unloading and handling activities at each customer location is given as well. The objective is to optimally choose gas stations and refueling amounts so as to minimize fuel costs. We refer to this problem as the sequence vehicle refueling problem with time windows.
3. The sequence vehicle refueling problem with time windows

3.2. Outline

An overview of the existing literature dealing with vehicle refueling problems is given in Section 3.3. The basic problems consider vehicle refueling between a pair of origin and destination locations, some of them integrating the route selection, others starting with a fixed path to be traveled between an origin and a destination. Depending on the problem setting, different graph structures are considered in the literature. These are analyzed in Section 3.4 and a choice is made for the particular problem that we study. In Section 3.5, we propose a MILP model for the sequence vehicle refueling problem with time windows. Note that in contrast to Chapter 2 and to Chapter 4, in this chapter we consider one hard time window per customer location. This is done for consistency reasons within this chapter to be able to easily integrate the proposed MILP model into the classical VRPTW which is done in a short digression in Section 3.7. Section 3.6 shows how gas stations can be mapped into the main route to achieve the graph structure proposed in Section 3.4. Numerical results that show the impact of price variations on the tour length when simultaneously planning vehicle routes and refueling are presented in Section 3.7. Note that Sections 3.3 and 3.4 are based on the research conducted in Bernhardt et al. (2017) and Sections 3.5 to 3.7 on the research performed in Bousonville et al. (2011).

3.3. Literature review

In the literature, different refueling problems have been analyzed. Besides studies for the road transportation sector, there are works that deal with refueling problems in railroad networks, the airline industry and maritime transportation. The positioning of fueling facilities is also a field of research.\(^{48}\)

In the following review, we consider studies that concentrate on road transport and vehicle refueling problems that include the identification of gas stations to be visited and the amounts of fuel to be purchased. For other modes of transportation we refer to Suzuki and Dai (2013). Routes may be given in advance or chosen together with the refueling strategy.

In the problem considered by Lin et al. (2007), the vehicle traverses a series of gas stations with different fuel prices while traveling along a fixed route. Detours to gas stations are ignored. At each gas station a decision has to be made on how much to refuel. The goal is to reach the destination with minimum total fuel cost. Lin et al. (2007) relate this problem to the inventory-capacitated lot-sizing problem and propose a linear-time greedy algorithm. The idea is to fill at each gas station just enough to reach the next cheaper gas station, or to fill up the tank if no cheaper gas station is reachable even with a full tank.

Lin (2008b) deals with the problem of finding an optimal refueling policy in a transportation network with fixed start and target vertices. The other vertices are gas stations with

\(^{48}\) The interested reader is referred to Suzuki (2008) and references therein.
3.3. Literature review

different fuel prices and other locations such as cities, suppliers or customers that may be but do not have to be visited. The goal is to find the cheapest path in the transportation network along with the corresponding refueling quantities without running out of gas. The start and end fuel levels may be arbitrary between a minimum fuel level (reserve quantity to remain in the tank at all times) and the tank capacity. Lin (2008b) takes all possible integer fuel levels per stop into consideration and proposes a polynomial time dynamic programming algorithm to solve the problem which depends on the number of vertices and the difference between the minimum fuel level and the tank capacity.

Khuller et al. (2007) consider several different refueling problems. One of them is the same problem addressed by Lin et al. (2007). Another one is the "gas station problem", which is similar to the problem addressed by Lin (2008b) with the difference that the end fuel level is set to be equal to the minimum fuel level. Khuller et al. (2007) present a different dynamic programming recursion to solve the problem. For the all-pairs version, a faster algorithm is proposed. Both algorithms run in polynomial time. Khuller et al. (2007) also study the "tour gas station problem" where a set of cities has to be visited in arbitrary order in a minimum cost tour. There may be cities with gas stations but some cities may not have a gas station. Gas stations located outside of cities may also be visited for refueling. Khuller et al. (2007) first concentrate on the uniform cost case where fuel prices are the same everywhere. Under certain assumptions\(^{49}\), the "uniform cost tour gas station problem" can be reduced to the TSP and can be solved with standard techniques. Building upon the Christofides heuristic for the TSP, the authors develop an approximation algorithm for the more general problem where for each city to be visited there is a gas station within a specified distance and requiring a fuel consumption of less than a half of the tank capacity. This algorithm is used within the heuristic for the "tour gas station problem" with arbitrary fuel prices. In the "sequence gas station problem", a cheapest way from a source to a final destination has to be found in the transportation network, visiting a set of locations in a given order. This problem can be reduced to the original gas station problem. The technique used will be discussed later when we consider the different possible graph structures (see Section 3.4). Finally, Khuller et al. (2007) consider the "single gas station problem" where the vehicle starts from the gas station and always has to return to it before it runs out of gas while visiting a number of cities.

Lin (2008a) considers a refueling problem that is similar to the gas station problem addressed by Lin (2008b). By analyzing the structure of optimal refueling policies, the problem is reduced to the classical shortest path problem. For this purpose, a transition graph is derived from the original graph, modeling all extremal transitions between gas stations where the vehicle arrives with the lowest fuel level allowed and gas stations that are left with a full tank. A corresponding distance measure that represents the transition cost is introduced. Lin (2008a) presents an algorithm that is faster than the one proposed by Khuller et al. (2007) for the all-pairs version. In addition, on the basis of the all-pairs version, he gives a solution method for the single-pair case with given end fuel level that may differ from the minimum fuel level.

Lin (2016) shows how to efficiently maintain and update routing and refueling information to be able to determine an optimal refueling strategy in quadratic time depending on the

\(^{49}\)It is supposed that every city has a gas station and the largest distance between any two cities is less than or equal to the tank capacity. No additional gas stations are considered, i.e. the set of gas stations is equal to the set of cities to be visited.
number of gas stations \( (n) \). With the help of shortest path trees and the usage of the transition network described in Lin (2008a), important routing information is determined in \( O(n^3) \) time using quadratic space which also depends on \( n \).

Suzuki (2008) refers to software products already in use by transport companies in the United States to plan vehicle refueling. He develops a mathematical programming model that mimics the behavior of standard fuel-optimizer software such as ProMiles, Expert Fuel, Fuel and Route, or Fuel Advice. The model can be used to optimally plan refueling stops at gas stations along with refueling quantities for a given route considering detour distances to and from gas stations. He stresses additional parameters that are taken into account by most fuel optimizer packages that are important in practice. These include, for example, the detour distances to gas stations or the availability of certain amenities to be able to eliminate unattractive gas stations that are far off the route or that do not have shower facilities. A minimum purchase quantity allows to control the frequency of refueling stops. A limitation to "network" gas stations (i.e. gas stations with purchase contracts) is also taken into account. Suzuki (2008) identifies the shortcoming with respect to other non-fuel cost elements that are interconnected with out of route miles to gas stations and the frequency of refueling stops such as vehicle deprecation cost and vehicle maintenance and opportunity costs in standard fuel optimizer software. He also stresses the underestimation of cost elements such as fuel consumption rates on non-highway roads that are of special importance if highways are left and detours to visit cheap gas stations are accepted. In Suzuki (2008), a MILP formulation is introduced to include such cost components in order to minimize the total cost of operating a vehicle in a given route. Numerical results for randomly generated instances are presented that compare the fuel purchasing cost and the total vehicle operating cost for the solutions obtained with the standard fuel optimizer model with those of the extended version. As a solution method, the simplex algorithm in conjunction with the branch-and-bound method is used.

Suzuki (2009) addresses a refueling problem that differs from the ones discussed so far. Usually, refueling strategies applied by standard-optimizer software deal with the questions which gas stations to choose and how much fuel to purchase. Suzuki (2009) mentions that transport companies are reluctant to introduce fuel optimizer software as they are afraid to suffer from limited actual cost savings because of low driver compliance rates and they even fear that drivers may move to other companies. He proposes a method that considers fuel price fluctuations over time and allows drivers to freely choose the gas stations they wish to visit. It is assumed that drivers take their daily rest period at a parking area of a truck stop where they also refuel. A corresponding refueling policy comprises the decisions on whether to refuel before or after taking a daily rest period at a truck stop chosen by the driver and on the refueling quantity. The latter may be equal to the minimum purchase quantity or the amount needed to fill the tank completely. Expected future prices at subsequent gas stations are taken into account by the stochastic dynamic programming model proposed. To predict future fuel prices at truck stops the OPIS (2017) database which provides fuel price information for truck stops in the U.S. and Canada is used. Computational results for randomly generated test instances are presented, comparing the costs for the case of random refueling behavior with those obtained when using the standard fuel optimizer model and those provided by the method proposed. For several scenarios, not only fuel costs but also driver compliance rates and driver replacement costs are taken into account. Although the lowest fuel cost is attained for the standard fuel optimizer model, under
certain conditions, the overall cost savings are higher with the proposed method.

Suzuki and Dai (2013) consider the vehicle refueling problem in combination with the route selection and propose a corresponding bicriteria MILP model. In contrast to Lin (2008a,b) and Khuller et al. (2007), the presented transportation network comprises vertices solely incorporated for the route selection subproblem. Gas stations are considered between each pair of those vertices in a similar way as it is done by Suzuki (2008, 2009) for the fixed path refueling problem. Additional constraints involve a limit on the maximum number of refueling stops and a limit on the maximum route duration. The duration for refueling is considered to be constant. Suzuki and Dai (2013) emphasize that it is important to consider both fuel costs and vehicle miles, and thus also integrate pollutants emission caused by increased fuel consumption into the decision process. The authors propose an optimization technique that involves the usage of a commercial optimization solver to construct the Pareto front. Different strategies are proposed to select the final solution according to the user preferences.

Suzuki (2014) outlines that there is no efficient algorithm in the literature that can solve the complex fixed-route vehicle refueling problem to optimality taking into account a minimum refueling quantity as well as detour distances to gas stations. He suggests the use of a preprocessing heuristic to eliminate gas stations that are guaranteed not to be chosen for refueling in the following solution process. For 16 instances based on real data provided by a fuel optimizer vendor, the variable-reduction technique removed between 46.9% and 60.1% of the gas stations. On average, this reduced the run time to about one-fourth of the original time needed to find an optimal solution. Suzuki (2014) also considers the quality of solutions determined by the heuristic used in the software of the fuel optimizer vendor. For the instances considered, the mean difference between the optimal solution and the one determined by the heuristic method was 0.3%. In some of the solutions produced by the software of the fuel optimizer vendor, less than the minimum purchase quantity was refueled at gas stations implying that the minimum purchase quantity is considered as a soft constraint.

The weight of the fuel in the tank as a variable part of the overall weight of the vehicle has an influence on the fuel consumption. Suzuki et al. (2014) aim at incorporating this weight as a factor for refueling decisions modifying the standard fuel optimizer model presented by Suzuki (2008, 2009) accordingly. Additionally, they consider the possibility to modify the minimum quantity of fuel to be left in the tank to not run out of fuel in case of unforeseen events depending on the gas station density that varies along the route. For the resulting nonlinear model, the authors propose a simple heuristic approach. To this end, they develop a relaxed MILP model based on the standard fuel optimizer model. By adding a penalty term in the objective function, the portion of the fuel tank that is never used is rewarded. The minimum fuel level is set per route segment. In their experiments, Suzuki et al. (2014) show the saving potential of their approach compared to the standard approach. They discover that in their experiments the overall fuel consumption is only reduced by up to 0.25%, whereas the savings in the overall fuel costs amount up to 1.74% compared with the standard approach presented in Suzuki (2008, 2009). This indicates that the reduction of the minimum fuel level for areas with a high gas station density is taken advantage of very extensively. At cheap gas stations, this allows to buy more fuel as, because of the reduced minimum fuel level, the fuel in the tank at arrival at a gas station may be less. Suzuki et al. (2014) also argue that the effectiveness of their approach may
improve as the maximum tank capacity increases. Based on Suzuki (2008), the authors also consider the impact of their approach on non-fuel cost. Since in their approach the detour distance and the frequency of refueling stops is decreased, the sum of other (direct and opportunity) costs associated with detour distances and durations, and the time needed for additional refueling stops is reduced as well.

Lin (2014) introduces two MILP models for vehicle refueling problems with route selection in a transportation network that is similar to the network considered by Lin (2008a,b). The models either minimize fuel cost or travel time, giving an upper bound on the overall travel time\textsuperscript{50} or the fuel cost, respectively, and thus only differ by the objective function and a single constraint that has to be chosen accordingly. The author proposes a formulation that restricts the solution space to only allow a simple path and then shows how to relax this condition.

Lin (2015) proves that the computational task to solve the MILP models presented in Lin (2014) is NP-complete, even if fuel prices do not vary or the fuel consumption and the travel time are linearly dependent. For these two cases the author proposes two polynomial-time approximation schemes.

Suzuki (2012) considers vehicle refueling in combination with the time-constrained single-vehicle routing problem (traveling salesman problem with time windows, TSPTW). He proposes a two-stage solution technique. In the first stage, the TSPTW is solved using a variant of the simulated annealing technique. Not only the best feasible tour is kept but also the M best feasible tours. In the second stage, for each tour chosen from a subset of the M tours determined in stage one, a MILP model is solved using the simplex algorithm in conjunction with the branch-and-bound method. The chosen subset depends on a customizable parameter. Similar to Bousonville et al. (2011), the MILP model is an extension of the standard fuel optimizer model with additional time window constraints. Strategies for improving the solving time are discussed. Numerical experiments for the proposed method are conducted for three real-word instances and a set of hypothetical instances (simulation experiments). The solution quality and the run time are compared to benchmark methods.

In the literature, no algorithms or models have been proposed so far that simultaneously plan vehicle refueling along with driver rest periods and breaks. In Chapter 4 we will present a MILP model to fill this gap. But before, in this chapter, we extend an existing MILP model to be able to consider time windows and a sequence of customer locations to be visited instead of a single origin and destination pair. The combined consideration of vehicle routing and refueling has not attracted much attention in the literature so far. A possible integration into the VRPTW will be presented in a short digression (see Section 3.7).

\textsuperscript{50} Note that this is equivalent to having a customer time window at the target location.
3.4. Graph structures

In the literature dedicated to refueling problems, different graph structures are proposed. Three of them are interesting for our objective and will be described in the following.

To determine the corresponding input parameters, at first a decision about the problem definition and the degree of abstraction has to be made. For two of the graph structures described in the following it is assumed that the choice of the optimal route to serve the customers has been made in advance. That means the sequence of stops to be visited is given with the input. In one of these graph structures the problem is reduced by neglecting detours to gas stations. The third graph structure builds upon the idea that for the choice of a route prices and locations of gas stations already play an important role. When choosing the route independently, gas stations along the route may be more expensive and detour distances to gas stations later chosen for refueling may be larger. This can be overcome by integrating the choice of gas stations into the process of finding an optimal route between consecutive customers. Therefore, the graph structure of this kind of problem represents a complete transportation network with vertices for customer locations and for gas stations, and arcs linking them.

We will now describe the three approaches in more detail and will explain their advantages and disadvantages. At the end of this section we will discuss the inclusion of routing decisions.

Lin et al. (2007) consider refueling along a fixed route. No detours to gas stations are taken into account. Adapted to the problem of finding an optimal refueling policy between an origin and a destination where no refueling may be allowed at the origin and destination\textsuperscript{51}, the resulting linear graph has \( n \) nodes representing the origin, destination and \( n - 2 \) gas stations (see the upper graph depicted in Figure 3.1). The origin may be the starting location for the vehicle at the beginning of the planning horizon or a customer vertex where loading and/or unloading of goods takes place. The destination may be the subsequent customer location or the final stop that should be reached at the end of the planning horizon. For a sequence of customer locations to be visited, the corresponding graph is shown next. The \( r \) different locations (origin, destination, customer locations and gas stations) are numbered from 0 to \( r - 1 \). It is simply the concatenation of origin and destination pairs and the linear graph structure remains.

The disadvantage of this graph structure is the underlying assumption that gas stations are always located on the route or at least extremely nearby such that detours to reach gas stations and to return to the route may be ignored. Gas stations located along highways are usually more expensive than stations that are a little farther away even if fuel cards are used. In addition, neglecting gas stations requiring a detour may reduce the solution space too much. On the other hand, considering gas stations with a detour but ignoring the detour distance will lead to solutions that are suboptimal in practice. As a detour to a gas station consumes time and fuel we actually want to know whether a price difference is worth a detour. Detours not considered in the planning phase may jeopardize the driver

\textsuperscript{51} In the original problem the route starts and ends at a gas station.
3. The sequence vehicle refueling problem with time windows

![Diagram of a linear graph with gas stations and customer locations labeled.]

Figure 3.1.: Linear graph

schedule and thus a punctual arrival at customer locations. Furthermore, they may lead to higher fuel expenditures than originally expected.

In Liu (2008a,b) and Khuller et al. (2007) (in their gas station problem), refueling in a transportation network is considered. In the basic problem, only the origin and destination vertices are set in advance, other vertices that represent gas stations or other locations may or may not be visited, and together with the origin and destination pair form a complete graph. The goal is to find the cheapest path from the origin to the destination. Extended by vertices for customer locations that have to be visited, the complete graph looks like the one depicted in Figure 3.2 in the upper part. To ensure that the sequence in which the customer locations have to be visited is kept, we need an extension. Khuller et al. (2007) call the underlying problem of finding the cheapest way starting from an origin to a final destination visiting a set of locations in a given order during the trip "the sequence gas station problem". For this task, the authors provide a corresponding graph structure. For each location in the sequence that is not equal to the origin or destination, a copy of the complete graph is made, i.e. if \( n \) is the total number of these locations, \( n - 2 \) copies are obtained. These graphs are joined by merging the equivalent to the \( i \)-th location (unequal to origin and destination) of the sequence from the \( i \)-th copy with the one from the \( (i+1) \)-th copy. If there is at least one customer location to be considered, the original graph is joined with the first copy by merging the first customer location and its equivalent in the copy. Figure 3.2 illustrates the case of one location that does not equal the origin or destination.\(^{52}\) Thus, the "sequence gas station problem" can be reduced to the "gas station problem", as a solution to the original problem can be obtained by finding an optimal path from the origin to the destination in the new graph.

One major drawback of this representation is the huge number of additional binary variables that have to be provided when describing the graph in a mathematical model. If \( n \) is the

---

\(^{52}\) Note that in Figure 3.2 a consecutive numbering of all locations, including gas stations, is chosen. The vertex with number 2 represents the first location of the sequence of locations that have to be visited.
number of locations to be visited, including the origin and destination, and \( m \) is the number of gas stations, the whole graph has \( (n - 1)(n + m)(n + m - 1) \) arcs. If in our mathematical model each arc is connected with the decision on whether or not to use this arc in the solution, we obtain for the case of only 10 locations to be visited in a sequence including origin and destination and 20 gas stations 7830 binary decision variables. Additional decision variables that are relevant to model rest periods and breaks on arcs have to be considered for each arc in the combined problem. However, there are additional reasons against this graph structure. In practice, an optimal route between locations is chosen based on different criteria, not only depending on refueling costs. Relevant aspects are, for example, the driving duration, toll costs, suitability of the streets for trucks, durations of border controls, and regional holidays with driving bans. Therefore, it is not a good idea to base the choice of routes between customer locations solely on the selection of gas stations. As long as other aspects that need to be taken into account cannot be formalized in a sufficient way and integrated in the routing decision, it is better to directly plan the routes between customer locations and try to stick to them as much as possible.

The mathematical models presented by Suzuki (2008, 2009) build upon the standard fuel optimizer model. The underlying idea of this model is that drivers follow the route previously determined (for example by a routing algorithm and then modified to also take
care of the aspects described above) and only leave it for refueling to return to the route afterward (see the upper graph in Figure 3.3).

![Linear graph supplemented by detour arcs](image)

Figure 3.3.: Linear graph supplemented by detour arcs

To simplify the problem, it is assumed that the route is left for refueling and entered after refueling at the same position and that the detour distances to and from the gas station are the same. In reality, this may not always be the case, especially if the highway is left for refueling. However, this graph structure allows us to consider detours without considering complete subgraphs for refueling. The original route is kept and only needs to be extended by detours. If a decision variable indicates that a gas station is chosen for refueling, the consumption for the detour to the gas station is added to the consumption for the path between the preceding location (gas station, customer location, origin or destination) and this gas station. The path between this gas station and the next location is extended by the detour from the gas station back to the route. Due to the reasons described above and the drawbacks of the other graph structures, we decided to use this base graph structure in the following.

The extended graph for a sequence of customer locations is depicted in Figure 3.3 at the bottom. Note that in contrast to Suzuki (2008) we allow detours to and from gas stations to differ.\textsuperscript{53} A consecutive numbering is chosen for all locations, i.e. for origin, destination, customer locations and gas stations, not differentiating between the kind of location.

Observe that if a gas station can be visited between several consecutive customer locations, it is listed for each of those pairs. Thus, depending on the time windows, it can be decided between which pair(s) of customer locations the gas station shall be visited if it is chosen for refueling.

\textsuperscript{53} This is for example used in Section 3.7 to map gas stations into the main route.
3.5. Mathematical formulation

Before presenting the new MILP model, input parameters and decision variables are introduced in Table 3.1 and Table 3.2, respectively.

The set of all locations is given by $S_{locations}$. To differentiate between location types, $S_{customers}$ denotes the set of all customer locations and $S_{stations}$ denotes the set of all gas stations. The origin and destination are mapped by the first vertex 0 and the last vertex $r - 1$, respectively.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{locations}$</td>
<td>Set of all locations (customer locations, gas stations, origin and final destination)</td>
</tr>
<tr>
<td>$S_{customers}$</td>
<td>Set of all customer locations that have to be visited; $S_{customers} \subset S_{locations}$</td>
</tr>
<tr>
<td>$S_{stations}$</td>
<td>Set of all gas stations that have been mapped into the route; $S_{stations} \subset S_{locations}$</td>
</tr>
<tr>
<td>0, $r - 1$</td>
<td>Origin and final destination, respectively$^{54}$, with ${0, r - 1} \subset S_{locations}$</td>
</tr>
<tr>
<td>$[TW_{\text{begin}}, TW_{\text{end}}]$</td>
<td>Time window at customer location $i \in S_{customers}$</td>
</tr>
<tr>
<td>$\Delta_{dr}^{(i,i+1)}$</td>
<td>Driving time needed to travel from $i$ to $i + 1$, $i = 0, \ldots, r - 2$ not including the time needed for out of route distances to and from gas stations</td>
</tr>
<tr>
<td>$\Delta_{drTo}^i$</td>
<td>Driving time needed to travel from the point of departure to the corresponding gas station $i$ (equals 0 if $i \notin S_{stations}$)</td>
</tr>
<tr>
<td>$\Delta_{drFrom}^i$</td>
<td>Driving time needed to travel from the gas station $i$ to the corresponding point of return (equals 0 if $i \notin S_{stations}$)</td>
</tr>
<tr>
<td>$\Delta_{cons}^{(i,i+1)}$</td>
<td>Fuel consumption when traveling from $i$ to $i + 1$, $i = 0, \ldots, r - 2$, not including the consumption for out of route distances to and from gas stations</td>
</tr>
<tr>
<td>$\Delta_{consTo}^i$</td>
<td>Fuel consumption when traveling from the point of departure to the corresponding gas station $i$ (equals 0 if $i \notin S_{stations}$)</td>
</tr>
<tr>
<td>$\Delta_{consFrom}^i$</td>
<td>Fuel consumption when traveling from the gas station $i$ to the corresponding point of return (equals 0 if $i \notin S_{stations}$)</td>
</tr>
<tr>
<td>$\Delta_{\text{refuel}}$</td>
<td>Time needed for refueling</td>
</tr>
<tr>
<td>$T_{\text{min}}$</td>
<td>Lower bound fuel, i.e. minimum amount of fuel to be maintained in the tank at all times</td>
</tr>
<tr>
<td>$T_{\text{min}}$</td>
<td>Minimum amount of fuel to purchase at a gas station</td>
</tr>
<tr>
<td>$T_{\text{max}}$</td>
<td>Vehicle tank capacity</td>
</tr>
</tbody>
</table>

$^{54}$ For the VRPTW considered in Section 3.7 the vertices 0 and $r - 1$ both denote the vehicle depot as start and end locations of each route.
3. The sequence vehicle refueling problem with time windows

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta_{i}^{\text{service}}$</td>
<td>Service time at customer location $i \in S_{\text{customers}}$</td>
</tr>
<tr>
<td>$P_{i}$</td>
<td>Fuel price at gas station $i \in S_{\text{stations}}$ (per unit of fuel)</td>
</tr>
<tr>
<td>$f_{\text{start}}, f_{\text{end}}$</td>
<td>Amount of fuel in the vehicle tank at origin 0 (start fuel level), respectively at destination $r - 1$ (end fuel level)</td>
</tr>
</tbody>
</table>

Table 3.1.: Input parameters

Observe that we assume that it is only worth to stop for refueling if at least a meaningful quantity specified by $\Delta_{\text{min}}$ is purchased.

Table 3.2 gives an overview of the decision variables of the new MILP model.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{i}^{\text{refuel}}$</td>
<td>Is equal to 1 if $i \in S_{\text{stations}}$ is chosen for refueling and 0 otherwise.</td>
</tr>
<tr>
<td>$\Delta_{i}^{\text{refuel}}$</td>
<td>Amount of fuel to purchase at gas station $i \in S_{\text{stations}}$</td>
</tr>
<tr>
<td>$T_{i}$</td>
<td>Remaining fuel in the tank upon arrival at location $i \in S_{\text{locations}}$ and before buying fuel</td>
</tr>
<tr>
<td>$\text{start}_{i}$</td>
<td>Begin of service at vertex $i \in S_{\text{customers}}$</td>
</tr>
<tr>
<td>$\Delta_{i}^{\text{work}}$</td>
<td>Duration of working activities associated with loading and/or unloading goods or refueling at vertex $i$</td>
</tr>
</tbody>
</table>

Table 3.2.: Decision variables

The mathematical formulation is as follows:

Minimize $\sum_{i \in S_{\text{stations}}} P_{i} \cdot \Delta_{i}^{\text{refuel}}$ \hspace{1cm} (3.5.1)

subject to:

$T_{i} \geq T^{\text{min}} \quad \forall i \in S_{\text{locations}}$ \hspace{1cm} (3.5.2)

55 Service time is the time needed at a customer location to fulfill a special service. Following the requirements of our main research goal which is formulated in the context of long haul freight transportation, the duration of loading and/or unloading at the corresponding location is addressed and other handling activities are included. In conjunction with the VRPTW (see Section 3.7) typically we either only consider loading activities or unloading activities at customer locations $i \in S_{\text{customers}}$, but not both. As there are different real-world applications for the VRPTW, in this context the service time can also refer to, for example, the duration of technical service, repair and maintenance or mobile sales.
Another or additional option would be to restrict the number of stops to a predefined maximum number which
3.5. Mathematical formulation
The objective function (3.5.1) minimizes the total cost of purchasing fuel. The refueling constraints (3.5.2) to (3.5.9) reflect the two following decisions:

- where to refuel (i.e. determination of refueling locations) and
- how much to refuel (i.e. determination of refueling quantities).

The amount of fuel in the tank, either at gas station \( i \) before purchasing fuel if gas station \( i \) is chosen for refueling (\( \Delta_{i}^{\text{refuel}} = 1 \)) or at the corresponding leaving point (if \( \Delta_{i}^{\text{refuel}} = 0 \), is denoted by \( T_i \). Constraints (3.5.2) ensure that the amount of fuel in the tank never falls below the defined reserve quantity \( T_{\text{min}} \). Constraint (3.5.3) sets the start fuel level \( f_{\text{start}} \) for the origin location 0. The minimum amount of fuel to be left in the tank at the final destination (\( f_{\text{end}} \)) is imposed by (3.5.4). Constraints (3.5.5) state that in case refueling takes place at gas station \( i \), the purchased amount \( \Delta_{i}^{\text{refuel}} \) has to be at least as much as the minimum purchase quantity \( \bar{\Delta}_{\text{min}} \). These constraints serve to raise the acceptance of drivers as they may not be willing to stop frequently for refueling very small amounts.\(^{56}\)

\[ T_0 = f_{\text{start}} \]  
\[ T_{i-1} \geq f_{\text{end}} \]  
\[ \Delta_i^{\text{refuel}} \geq \bar{\Delta}_{\text{min}} \alpha_i^{\text{refuel}} \quad \forall i \in S_{\text{stations}} \]  
\[ \Delta_i^{\text{refuel}} \leq T_{\text{max}} \alpha_i^{\text{refuel}} \quad \forall i \in S_{\text{locations}} \]  
\[ \Delta_i^{\text{refuel}} \leq T_{\text{max}} - T_i \quad \forall i \in S_{\text{stations}} \]  
\[ \alpha_i^{\text{refuel}} = 0 \quad \forall i \in S_{\text{customersons}} \cup \{0,r-1\} \]  
\[ T_{i+1} = T_i + \Delta_i^{\text{refuel}} - \bar{\Delta}_{\text{consFrom}} \alpha_i^{\text{refuel}} - \bar{\Delta}_{\text{consTo}} \alpha_{i+1}^{\text{refuel}} \] 
\[ \forall i \in S_{\text{locations}} \setminus \{r-1\} \]  
\[ \bar{\Delta}_{\text{from}} = \Delta_i^{\text{service}} + \Delta_i^{\text{refuel}} \alpha_i^{\text{refuel}} \quad \forall i \in S_{\text{locations}} \]  
\[ \text{start}_{i+1} \geq \text{start}_i + \Delta_i^{\text{work}} + \bar{\Delta}_{\text{drFrom}} \alpha_i^{\text{refuel}} + \bar{\Delta}_{\text{drTo}} \alpha_{i+1}^{\text{refuel}} \] 
\[ \forall i \in S_{\text{locations}} \setminus \{r-1\}, i+1 \in S_{\text{customers}} \]  
\[ \text{start}_{i+1} = \text{start}_i + \Delta_i^{\text{work}} + \bar{\Delta}_{\text{drFrom}} \alpha_i^{\text{refuel}} + \bar{\Delta}_{\text{drTo}} \alpha_{i+1}^{\text{refuel}} \] 
\[ \forall i \in S_{\text{locations}} \setminus \{r-1\}, i+1 \in S_{\text{locations}} \setminus S_{\text{customers}} \]  
\[ \text{start}_i \geq \bar{TW}_i^{\text{begin}} \quad \forall i \in S_{\text{customers}} \]  
\[ \text{start}_i \leq \bar{TW}_i^{\text{end}} \quad \forall i \in S_{\text{customers}} \]  
\[ \alpha_i^{\text{refuel}} \in \{0,1\} \quad \forall i \in S_{\text{locations}} \]  
\[ \Delta_i^{\text{refuel}} \geq 0 \quad \forall i \in S_{\text{locations}} \]  
\[ T_i \geq 0 \quad \forall i \in S_{\text{locations}} \]  
\[ \Delta_i^{\text{work}} \geq 0 \quad \forall i \in S_{\text{locations}} \]  
\[ \text{start}_i \geq 0 \quad \forall i \in S_{\text{locations}} \]  
\[ f_{\text{start}} \]

\(^{56}\) Another or additional option would be to restrict the number of stops to a predefined maximum number which is, for example, dependent on the original length of the complete route. Note that the third objective function that is used in the last optimization step for the combined model described later penalizes the number of gas
In addition, they are important if minimum purchase quantities are necessary to get a discount. If no refueling takes place at location \( i \), the refueling quantity is set to be 0 by constraints (3.5.6). Taking into account the maximum tank capacity, inequalities (3.5.7) ensure that the refueling quantity \( \Delta_{i}^{\text{refuel}} \) at \( i \) is equal to or less than the tank capacity minus the amount of fuel in the tank \( T_{i} \) when reaching gas station \( i \).

By constraints (3.5.8), the variables that indicate if refueling takes place at location \( i \) are set to be 0 for all locations that do not represent a gas station.

The standard fuel optimizer model described by Suzuki (2008, 2009) considers linear consumptions by suggesting the use of consumption rates per distance unit. In contrast to this, we consider individual fuel consumptions per arc \( (\bar{\Delta}_{(i,i+1)}^{\text{cons}}) \), i.e. for each pair of consecutive locations, to be able to integrate more precise consumption data relying, for example, on road topography, route types and/or empirical values if available. Also for detours individual fuel consumptions are possible. Deviating from the standard fuel optimizer model, we assume that the fuel consumption from the point where the route is left to head for the gas station, \( \bar{\Delta}_{i}^{\text{consTo}} \), may differ from the fuel consumption for the way back, \( \bar{\Delta}_{i}^{\text{consFrom}} \).

The remaining quantity of fuel upon arrival at each location (i.e. gas station, customer location or final destination) is determined by constraints (3.5.9) which are adopted from the standard fuel optimizer model. The constraints are customized to individual consumptions and different detour consumptions depending on whether heading for the gas station or returning to the route. This quantity depends on the quantity available in the tank at the previously visited location, on the quantity purchased at the previous location and on the fuel consumed during the trip to reach the current location. If the previously visited location or the current one is a gas station then the fuel consumed due to the detour is also taken into account.

As a vertex may represent a customer location or a gas station, there are two different kinds of working activities that may take place: working activities associated with loading and/or unloading or refueling. Both can be treated similarly as far as time aspects are considered. It is important to know the estimated duration of the working activity. A new variable \( \Delta_{i}^{\text{work}} \) is introduced that represents the duration of the working activity at location \( i \) (see (3.5.10)). It is composed of the time dedicated to refueling \( \Delta_{i}^{\text{refuel}} \) and the time for loading and/or unloading the vehicle \( \Delta_{i}^{\text{service}} \). The working time for refueling is set to be zero for all non gas station locations. For reasons of simplicity it is assumed to be constant for each gas station and it is only taken into account if the corresponding gas station \( i \) is chosen for refueling (\( \alpha_{i}^{\text{refuel}} = 1 \)). The working time for loading and/or unloading goods is set to be zero for all non-customer locations.

Constraints (3.5.11) set lower bounds on the beginning of service times (that is, for example, the beginning of loading and/or unloading) at every location. Their derivation is similar to the one that is described for constraints (3.5.9), now considering durations instead of consumptions. Note that if \( i + 1 \) is not a customer location, \( \text{start}_{i} \) denotes the arrival time at the corresponding location which is given by (3.5.12).

Constraints (3.5.13) and (3.5.14) guarantee that the time windows are satisfied. If a vehicle arrives at a customer location before the beginning of the corresponding time window then
it has to wait. In this case, the related constraint from (3.5.11) is satisfied as a strict inequality. As there are no time windows at non-customer locations, no waiting times are considered in (3.5.12).\(^{57}\) In contrast to Chapter 2 and Chapter 4 we only consider one time window per customer location and do not allow lateness by introducing hard time window constraints. Constraint (3.5.15) initializes the time at the origin location.

Finally, constraints (3.5.16) to (3.5.20) represent binary and non-negativity conditions. Observe that constraints (3.5.17) are redundant.

In the next section, we will describe the mapping of gas stations into the main route to obtain the graph structure chosen in Section 3.4. This mapping was used to prepare the test instances that will be presented in Section 3.7 in which the possible integration of the MILP model for the fixed sequence refueling problem with time windows into the VRPTW is explored in a short digression. Afterward, in Chapter 4, we will return to our main research topic merging the MILP model developed in this section with the MILP model(s) described in Chapter 2, and showing the necessary changes to be able to merge the two models.

### 3.6. Mapping gas stations into the main route

A graph is required that includes vertices for the origin, the destination, all customer locations to be visited by the vehicle as well as the gas stations that are potentially attractive for a refueling stop (e.g. all gas stations within 20 km of the main route). In Section 3.4 we chose a graph structure that allows a comfortable and efficient modeling. For this graph structure, we assume that the point where the route is left to head for a gas station equals the point where it is entered after refueling. Unlike Suzuki (2008, 2009), we allow detours to and from gas stations to differ.

In the following, let \(c\) and \(c + 1\) be two consecutive customer locations. If \(c + 1\) denotes the first customer location, then \(c\) is the origin and if \(c\) marks the last customer location, then \(c + 1\) is the final destination. Let \(i\) be an arbitrary gas station that is considered for refueling between visiting \(c\) and \(c + 1\). If \(i\) is chosen for refueling and \(i\) does not lie on the direct path between \(c\) and \(c + 1\), the vehicle has to divert from the route at some point \(a\) and to re-enter the route at some point \(b\) (see Figure 3.4).

In reality, \(a\) and \(b\) do not necessarily have to be equal. It may even be the case that the way back to the route has to differ from the way to the gas station as one-way streets are involved or a highway exit lies apart from the next possible access. Depending on the length of detour distances, this can be relevant. Just taking the overall detour distance to reach the gas station and to return back to the route and dividing it by two to obtain the one-way distance may yield a distorted fuel level assumption for the arrival at a gas station. Thus, the refueling quantities determined may not be accurate. In the following,

---

\(^{57}\) Note that a consideration of the constraints (3.5.11) for non-customer vertices and removing constraints (3.5.12) is possible. In this case, waiting time that is induced by one time window may spread over several arcs. The solution space would increase.
we will show a possibility for mapping gas stations into the main route to obtain the graph structure chosen in 3.4 considering different detours from and to gas stations.

For the proposed mapping, the distances $\overline{\text{dist}}_{(c,c+1)}$ between each pair of consecutive customer locations including origin and destination, $c$ and $c+1$, need to be known. For each such pair of locations $c$, $c+1$ with the above properties, the direct distances from $c$ to all attractive gas stations $i$, $\overline{\text{dist}}_{(c,i)}$, as well as the direct distances between all such gas stations $i$ to $c+1$, $\overline{\text{dist}}_{(i,c+1)}$, are necessary input parameters for the mapping as well. Figure 3.5 depicts the original distances. Note that if fuel consumptions and driving durations are not assumed to be linearly dependent on the corresponding distances, analogous information on consumptions and durations is necessary.

If the vehicle deviates from the direct path from $c$ to $c+1$ to visit a gas station $i$ then the travel distance increases by $\overline{\text{dist}}_{(c,i)} + \overline{\text{dist}}_{(i,c+1)} - \overline{\text{dist}}_{(c,c+1)}$. This overall detour distance needs to be preserved.

To construct the new graph, every gas station $i$ between two consecutive locations $c$ and $c+1$ is mapped into the direct path, thus creating a new vertex $i'$. The corresponding detour distances $\Delta_i^{\text{distTo}}$ and $\Delta_i^{\text{distFrom}}$ are determined accordingly. The mapping procedure considers three different cases. The intermediate target structure after the transformation described for each of these cases is depicted in Figure 3.6. The total distance traveled by a vehicle during its trip from $c$ to $c+1$ that visits a gas station $i$ is given by

$$\overline{\text{dist}}_{(c,i')} + \Delta_i^{\text{distTo}} + \Delta_i^{\text{distFrom}} + \overline{\text{dist}}_{(i',c+1)}.$$
Case 1: $\overline{\text{dist}}_{(c,i)} < \overline{\text{dist}}_{(c,c+1)}$ and $\overline{\text{dist}}_{(i,c+1)} < \overline{\text{dist}}_{(c,c+1)}$

This situation is already reflected in Figure 3.5. As shown in Figure 3.7, the gas station $i$ is represented by vertex $i'$ which is inserted between $c$ and $c+1$.

The following distances are assigned to the new arcs:

$$
\overline{\text{dist}}_{(c,i')} = \overline{\text{dist}}_{(c,i)}, \quad \overline{\text{dist}}_{(i',c+1)} = \overline{\text{dist}}_{(c,c+1)} - \overline{\text{dist}}_{(c,i)}
$$

If no refueling takes place between $c$ and $c+1$ and therefore, gas station $i$ is not visited then the distance traveled by the vehicle is equal to $\overline{\text{dist}}_{(c,c+1)}$. In contrast, if the vehicle stops at gas station $i$ then a detour distance has to be taken into account in addition to the distances given in (3.6.1):

$$\overline{\Delta}_i^{\text{distTo}} = 0, \quad \overline{\Delta}_i^{\text{distFrom}} = \overline{\text{dist}}_{(c,i)} + \overline{\text{dist}}_{(i,c+1)} - \overline{\text{dist}}_{(c,c+1)}$$

Hence, in this case the total distance traveled by the vehicle corresponds to

$$\overline{\text{dist}}_{(c,i')} + \overline{\text{dist}}_{(i',c+1)} + \overline{\Delta}_i^{\text{distTo}} + \overline{\Delta}_i^{\text{distFrom}} = \overline{\text{dist}}_{(c,i)} + \overline{\text{dist}}_{(i,c+1)}.$$
Case 2: $\overline{\text{dist}}(c,i) < \overline{\text{dist}}(c,c+1)$ and $\overline{\text{dist}}(i,c+1) \geq \overline{\text{dist}}(c,c+1)$

Figure 3.8 depicts this case. The projection of the gas station is indicated by a dashed arc and coincides with vertex $c$. As before, we denote by $i'$ the new vertex (which is omitted from the figure for the sake of clarity).

![Diagram](image)

Figure 3.8.: Mapping a gas station $i$ into the main path between customer locations $c$ and $c + 1$ (case 2)

The following distances and detours are assigned:

$$\overline{\text{dist}}(c,i') = 0, \quad \overline{\text{dist}}(i',c+1) = \overline{\text{dist}}(c,c+1)$$

As a result, a refueling stop at gas station $i$ again leads to the desired total distance:

$$\overline{\text{dist}}(c,i') + \overline{\text{dist}}(i',c+1) + \overline{\Delta}_{i}^{\text{distTo}} + \overline{\Delta}_{i}^{\text{distFrom}} = \overline{\text{dist}}(c,i) + \overline{\text{dist}}(i,c+1).$$

Case 3: $\overline{\text{dist}}(c,i) \geq \overline{\text{dist}}(c,c+1)$

As shown in Figure 3.9, the mapping results in a new vertex $i'$ that coincides with vertex $c + 1$ (the dashed arc indicates the projection).

![Diagram](image)

Figure 3.9.: Mapping a gas station $i$ into the main path between customer locations $c$ and $c + 1$ (case 3)
Moreover, the following distances and detours are defined:

$$
\frac{\tilde{d}_{i}^\text{distTo}}{\tilde{d}_{i}^\text{distFrom}} = \frac{\bar{d}_{(c,i)} - \bar{d}_{(c,c+1)}}{\bar{d}_{(i,c+1)}} = \frac{\bar{d}_{(i',c+1)}}{\bar{d}_{(i,c+1)}}
$$

Again, the total traveled distance from $c$ to $c + 1$ through $i$ is equal to

$$
\bar{d}_{(c,i')} + \bar{d}_{(i',c+1)} + \tilde{d}_{(c,i)} + \tilde{d}_{(i,c+1)} = \bar{d}_{(c,i')} + \bar{d}_{(i,c+1)}
$$

After having mapped all gas stations $i$ for the pair of non-gas station locations $c$ and $c + 1$, it is necessary to insert them in the direct path between $c$ and $c + 1$ in the appropriate order. This entails sorting the mapped gas stations in ascending order with respect to their distances to vertex $c$ (ties are broken arbitrarily). The sorted mapped gas stations are then inserted between $c$ and $c + 1$ and linked by arcs as displayed in Figure 3.10.

![Figure 3.10: Potential gas stations to be visited on the path from $c$ to $c + 1$](image)

The length of an arc $\tilde{d}_{(i_j,i_{j+1})}$ connecting two consecutive mapped gas stations $i_j$ and $i_{j+1}$ is given by $\bar{d}_{(c,i_j)} - \bar{d}_{(c,i_{j+1})}$. The distance between location $c$ and the first mapped gas station is given by $\tilde{d}_{(c,i_1)} = \bar{d}_{(c,i_1)}$ and the distance between the last mapped gas station in the sequence is $\tilde{d}_{(c,c+1)} = \bar{d}_{(c,c+1)} - \bar{d}_{(c,m)}$.\(^58\)

The above procedure is applied to every pair of consecutive customer locations in a given vehicle route. Note that if fuel consumptions and driving durations are not assumed to be linearly dependent on the route lengths, the above transformation has to consider the computation of the corresponding parameters.

Observe that the structure of the new graph preserves the original sequence in which customers are to be serviced. Finally, it should be emphasized that contrary to Suzuki (2008, 2009), the detour distances to reach a gas station and to return to the main path may not necessarily be the same. This means that the construction of the new graph takes into account that the point of return to the path may not coincide with the point of departure. This is an important aspect since in practice road distances instead of Euclidean distances are relevant. Hence, we are able to model realistic detours. However, if the vehicle visits two or more gas stations between two customers then the total traveled distance may not be accurate according to our graph. This is the case, for example, when the shortest

\(^58\) For reasons of simplicity the apostrophe referring to a mapped gas station is omitted for the distances between two consecutive locations on the main route.
path between two gas stations chosen consecutively does not go through the original route. This case rarely occurs in the Solomon test instances\textsuperscript{59} for the VRPTW that we consider in the following section due to the fact that the distances between two customers as well as the distances between a depot and a customer are not too large. Moreover, the tank capacity makes it almost always possible to reach a customer without having to stop for refueling more than once. Thus, the accuracy of the graph constructed for each one of the Solomon instances is acceptable.

**3.7. Integration of vehicle refueling into the VRPTW - A short digression**

Vehicle refueling can be considered in different contexts. In this thesis, we concentrate on problem settings in long-haul freight transportation integrating vehicle refueling into the scheduling of driver activities given a fixed sequence of customer locations to be visited. In the following digression, we propose an extension of the vehicle routing problem with time windows (VRPTW) making use of the MILP model developed in Section 3.5 and the mapping of gas stations into the main route presented in Section 3.6.

Similarly to the VRPTW, the planning tasks mentioned in Section 1.2 comprise the assignment of orders to vehicles (clustering), and the determination of the sequence of customer locations to be visited by each vehicle. In long-haul transportation, requests consist of a pickup and a corresponding delivery location, and both are not equal to the depot. However, the problem setting itself is related and this is an interesting field of research even without considering rest periods and breaks. Additionally, this section may deliver interesting input for future research to further address integration of vehicle refueling into the pickup and delivery problem with time windows (PDPTW) which is a generalization of the VRPTW with additional sequencing constraints.

In typical real-world applications of the VRPTW in road transport, distances are shorter than in long-haul freight transportation. Classical examples not only incorporate the delivery (or pickup) of goods to (or from) a set of customers but also other kinds of services such as the fulfillment of technical services, repair and maintenance or mobile sales. The consideration without integration of Regulation (EC) No 561/2006 is possible if, for example, daily tours are considered that do not comprise more than 9 hours of pure driving time per vehicle. A break of at least 45 minutes has to be planned if the driving duration exceeds 4.5 hours.\textsuperscript{60}

This digression is based on the assumption of a uniform fleet of vehicles all located at the same depot and having the same capacity. Routes are to be constructed to service a set of customers with given time windows and service times. We extend this well-known VRPTW (Cordeau et al. (2002), Golden et al. (2008)) by assuming that a vehicle has a

\begin{itemize}
  \item \textsuperscript{59} See Solomon (1987).
  \item \textsuperscript{60} It would, for example, be possible to plan a lunch break with a fixed start. In this case, time windows would be shifted forward accordingly to incorporate the break into the planning.
\end{itemize}
constant fuel consumption per distance unit. The quantity of fuel in the vehicle tank has to be above a minimum level (for safety reasons) and cannot exceed the tank capacity at all times. The objective function is decomposed into two parts. The first objective is to serve the customer requests with a minimal number of vehicles. The second objective is to minimize the total fuel cost (instead of the common criterion of minimizing the total distance). This means that, similarly as described by Suzuki (2012) who considers the single vehicle case (see Section 3.3), our approach does not aim at minimizing the overall fuel consumption but the overall cost for refueling. This does not necessarily mean that the shortest routes are chosen as unit fuel costs at gas stations also affect fuel expenditures.

In the following section, a heuristic developed to solve the VRPTW extended with refueling decisions is introduced. Numerical experiments and their results are then presented in Section 3.7.2.

### 3.7.1. A heuristic for the VRPTW with refueling

A variety of different heuristics have been proposed for the VRPTW (Bräysy and Gendreau (2005a,b)). Besides time windows and capacity constraints, we now have to make sure that a solution to the refueling problem exists. As a gas station usually will not be located directly on the route, this implies a detour. Obviously, the time needed for this detour and the time needed to refuel will make especially those base solutions with tight time windows infeasible. Hence, we cannot solve the problem sequentially by first running a VRPTW heuristic and then finding an optimal refueling strategy. As a consequence, using the model presented in the previous section we ensure feasibility each time a route is modified, e.g. a customer is added. Before constructing the refueling model for a given route we perform a preprocessing step to eliminate those gas stations that cannot be part of the solution. This is for instance the case if the detour to a gas station plus the refueling time would cause lateness for at least one of the subsequent customer time windows. The time for solving the model depends on various factors (we will analyze the impact of the route length on run time in Section 3.7.2). In any case, the embedded graph transformation and the call of the optimization solver to solve the refueling problem are relatively expensive. As local search requires an efficient evaluation of moves, this technique would be extremely time-consuming as well. The same also holds for Large Neighborhood Search (Pisinger and Ropke (2010)). Therefore, we limit our algorithmic approach to an integration of the fuel optimization model into a construction heuristic, namely the well-known Solomon I1 heuristic (Solomon (1987)). In this heuristic, the routes are constructed in a sequential manner. Starting with a customer location as a seed node for each new route, further customer locations are selected by a heuristic criterion. Once none of the remaining customer locations can be added to the current route - due to capacity or time window restrictions - a new route is started. Selecting a customer location (apart from the customer location for the seed node) is a two-step process. First, for each customer location the best insertion point is determined. This includes performing a feasibility test for each remaining customer location and each possible insertion point.

---

61 By a "base solution" we mean a solution with routes that only include customer locations and no intermediate stops at refueling points.
Second, the best customer location among the feasible insertions is chosen. As verifying if a feasible refueling solution exists is costly, we modify this approach slightly. In the first step we only test the classical feasibility (time windows, capacity) as before. Then, when selecting the best customer location we perform the feasibility check on refueling and select the best feasible one or continue with a new route if no feasible solution is found. The original heuristic can be parameterized for the selection of the seed node (customer location with earliest time window versus farthest away) as well as for the other customer locations (distance minimization versus shift of beginning of service at the next customer). Together with another parameter that controls the preference for customer locations far away from the depot, in total eight parameter combinations are possible and the best of the corresponding eight solutions is kept at the end. We keep this logic as well as the selection criteria.

3.7.2. Numerical experiments

The well-known Solomon benchmark instances (Solomon (1987)) for the VRPTW consist of 56 instances, distributed over 6 sets. The locations of all 100 customers are either clustered (sets C1 and C2), or randomly generated (sets R1 and R2), or include a mix of both types (sets RC1 and RC2). Each set contains between 8 and 12 instances. In test instances with a short scheduling horizon (sets C1, R1 and RC1) routes have approximately 5-10 customers. In contrast, a long scheduling horizon allows routes with more than 30 customers (sets C2, R2 and RC2). Furthermore, both tight and large time windows are considered. However, the percentage of customers with time windows varies between 25\% and 100\%. The Euclidean distance between two customers also corresponds to the travel time. We extended the Solomon test instances to include the additional information needed for the refueling model, i.e. the candidate locations for refueling, the associated fuel prices as well as the additional refueling parameters described in Table 3.1. As fuel consumption (as well as the travel time mentioned above) is assumed to be constant per distance unit, the price per liter can be mapped into a price per distance unit and therefore, the vehicle tank capacity can also be expressed in a maximum distance reach. The gas stations are positioned on an equally spaced grid between the point (0, 0) and the point (100, 100). For each of the 56 original test instances we derive two new versions, one in which the grid is spaced by 5 distance units and one where the grid is spaced by 10 units. In the first case, 441 gas stations are created, while the second case comprises 121 gas stations. Figure 3.11 shows an example of an instance with clusters of customers.
A gas station is located at each intersection of two grid lines. For the price differences between the gas stations we compare variations of 0, 10 and 20 percent. The price at each station is determined by a uniformly distributed random variable in the interval $[100, 100 + x]$ with $x$ denoting the price variation in percent. Recall that the price $P_i$ is given per distance unit traveled. Start and end fuel level, $f^{start}$ and $f^{end}$, are equally fixed to 30 distance units. This implies that the total refueling quantity equals the length of the overall route (including detours). Furthermore, the tank capacity $T^{max}$ is set to 60. As the mean length of a route over all instances (in the best known solutions (SINTEF (2018)) is about 200, a lower bound on the mean number of refueling stops per route is equal to 4. The parameters $T^{min}$ and $\Delta^{min}$ take the values 0 and 0.01, respectively. The time needed for refueling $\Delta^{refuel}$ is set to be equal to 2.

By combining the three price structures with the two versions of the gas station grid, we derived in total six variants of each original Solomon test instance. The following analysis refers to experiments performed on this set and on the resulting 336 solutions. All runs were executed on a PC with 2.53 GHz and 3 GB RAM. The implementation was done in Java. The refueling optimization problem described in Section 3.3 was solved with CPLEX 12.1.

Table 3.3 shows the results in terms of the resulting number of routes, tour length and fuel cost summed up over all 56 instances for the six parameter combinations.\(^{62}\)

\(^{62}\) Note that the results of our implementation differ slightly from those reported in the original paper by Solomon (1987). We use our results as a basis for the comparison because the solutions to our problems with refueling were obtained by relying on the same implementation.
Table 3.3.: Computational results for the 6 parameter sets; each line sums up the results of the 56 runs

<table>
<thead>
<tr>
<th>price variation in %</th>
<th>distance units gas station grid</th>
<th>total number of routes</th>
<th>sum of tour lengths</th>
<th>sum of fuel costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>without refueling</td>
<td>458</td>
<td>71,612.10</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>466</td>
<td>72,567.22</td>
<td>7,256,722.43</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>469</td>
<td>73,181.67</td>
<td>7,318,164.69</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>467</td>
<td>72,831.59</td>
<td>7,352,358.53</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>469</td>
<td>73,541.19</td>
<td>7,545,596.50</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>467</td>
<td>72,906.82</td>
<td>7,468,975.69</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>469</td>
<td>73,765.64</td>
<td>7,783,212.85</td>
</tr>
</tbody>
</table>

Not surprisingly, compared to the original situation without refueling, the total number of routes increases and this increase is higher for the instances with the wider gas station grid. A closer inspection of the problem sets reveals that the contribution stems mainly from the RC1 and R1 sets.

Table 3.4 reports the impact of price variation on the tour length. It turns out that the increase in tour length is quite moderate when increasing the price variation.

<table>
<thead>
<tr>
<th>price variation in %</th>
<th>distance units gas station grid</th>
<th>increase in tour length</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>0.36%</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.49%</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>0.47%</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.80%</td>
</tr>
</tbody>
</table>

Table 3.4.: Relative increase of the tour length when comparing a spread of 10%, respectively 20% to a constant price structure (no variations)

Observe that when gas stations are farther apart the vehicle may have to drive longer to reach refueling points with cheaper prices. Therefore, in test instances with a wider gas station grid the probability of obtaining longer routes is higher. However, the length increase does not seem to be very significant. In instances with tight time windows, detours for refueling are certainly restricted. In contrast, larger deviations from the main route are possible in those instances with wider time windows. As the results shown in Table 3.4 refer to average values, a balance between these two cases is achieved.

Although all Solomon test instances comprise 100 customers, the vehicle capacity is larger
in the C2, R2 and RC2 sets compared with the C1, R1 and RC1 sets. As a consequence, the resulting routes are longer and the size of the refueling problem is also larger. Figure 3.12 displays information on the run time behavior of the CPLEX optimization solver as a function of the route length.

Figure 3.12.: Run time for one instance depending on the mean number of customer locations per route in the solution of this instance

We plotted the total run time per instance against the mean number of customers per route in the final solution. The run times vary between 10.5 seconds and 2023.9 seconds. While the instances with few customers per route (on average at most 9 customers) are solved mostly in less than one minute, among the instances with many customers per route there are also cases with considerably longer run times. As in some test instances CPLEX was not able to find a feasible solution to the refueling problem after several hours, we imposed a time limit of 10 seconds. If no feasible refueling solution is found within this time limit, the next best insertion of a customer location is tried out. Table 3.5 contains information on the number of times CPLEX failed to find a solution within the given time limit per instance set. The number of fails represents a small percentage of all internal optimization solver calls. It is interesting to note however, that the sets that seem to contain the harder instances for the basic VRPTW (namely R2 and RC2) keep this characteristic when solving the refueling problem on a single route basis.

---

63 While constructing the routes obviously most of the calls to the optimization solver are performed with input routes of (much) smaller size.

64 E.g. the number of calls of the optimization solver for the set R2 is approximately 63,408.
<table>
<thead>
<tr>
<th>instance set</th>
<th>number of fails due to reaching time limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>71</td>
</tr>
<tr>
<td>R1</td>
<td>0</td>
</tr>
<tr>
<td>R2</td>
<td>387</td>
</tr>
<tr>
<td>RC1</td>
<td>0</td>
</tr>
<tr>
<td>RC2</td>
<td>239</td>
</tr>
</tbody>
</table>

Table 3.5.: Discarded inserts of customer locations due to having reached the time limit to obtain a feasible refueling solution
4. The combined problem

In Chapter 2, we considered the scheduling of driver activities in accordance to Regulation (EC) No 561/2006 and developed a MILP model for a fixed vehicle route and multiple soft time windows per customer location. In Chapter 3, the sequence vehicle refueling problem with time windows was introduced. We will now show how to merge the MILP models.\footnote{An integration of refueling decisions is possible for both models from Chapter 2, for the one that considers the optional rules and the one that ignores them. We will show the integration for the model with optional rules. Numerical experiments are only described for the MILP model resulting from this step. If optional rules are not considered, then constraints (4.2.14) and (4.2.15) can be dropped and the objective function (4.2.18) has to be modified. For reasons of simplicity, in the following, we talk about "the MILP model developed in Chapter 2" referring to the model which considers the optional rules.}

The starting point of the problem consists of an origin location, a pre-specified sequence of customer locations which have to be visited by the vehicle for loading and/or unloading goods in the current week and a final destination. Regulation (EC) No 561/2006 on rest periods and breaks should be taken into account. To be able to plan in a rolling horizon manner and to reschedule activities in case of unforeseen events, the driver status at the beginning of the planning process provides information about when the next daily or weekly rest period or break is necessary. Moreover, it also gives information on the usage of the optional rules. Thus, the planning may start at the beginning of the week but it is also possible to start on any day of the week in case the driver has already started to serve customer requests. For each customer location there are one or more time windows among which a choice has to be made. We do not consider customer locations with time windows that start after the presumed end of the next weekly rest period. This means that we do not schedule driver activities spreading over two weeks. Loading and/or unloading goods at customer locations should start within the time windows or opening hours (modeled as large time windows). An earlier start before the lower limit of a time window is prohibited. Lateness is only allowed if it cannot be avoided and is strongly penalized in the first objective function. This makes it possible to give feedback to the dispatcher in case that there does not exist a solution without lateness (see also Section 2.1). The time that is needed for loading, unloading and handling activities at each customer location is given as well. Geographical positions of gas stations along the route and the corresponding fuel prices are known as well as the current fuel level in the tank and the maximum tank capacity. Driving durations between consecutive locations and fuel consumptions are additional input parameters. For detour durations and consumptions separate parameters are available. Parameters like the minimum purchase quantity, the minimum fuel level to be maintained in the tank at all times as well as the end fuel level have to be chosen according to user preferences. The goal is to select time windows and gas stations, determine refueling quantities and plan driver activities that comply with Regulation (EC) No 561/2006 in such a way that lateness, overall schedule duration and fuel expenditures are minimized.
The simultaneous consideration of these tasks will reduce inefficiencies that arise from
the distributed decision making of drivers and dispatchers. As described in Section 1.2.2,
minimizing lateness and minimizing fuel costs are two conflicting goals which have to be
taken into account.

The MILP model that will be proposed builds on the MILP model with optional rules for
scheduling driving times, rest periods and breaks described in Chapter 2. For a review
of research on including regulations concerning rest periods and breaks in operational
transportation planning considering Regulation (EC) No 561/2006 see Section 2.4. A
review of the literature dealing with vehicle refueling problems has already been given in
Section 3.3. To the best of our knowledge, the combined problem has not been considered
in the literature so far.

4.1. Outline

The MILP model introduced in Bernhardt et al. (2017) serves as a basis for this chapter.
In Section 4.2 it is shown how to merge the MILP model described in Chapter 3 and the
MILP model with optional rules developed in Chapter 2 to simultaneously plan refuel-
ing, customer time windows and driver activities in accordance with Regulation (EC) No
561/2006. The solution process to solve the resulting multicriteria optimization problem
with the help of an optimization solver is described in Section 4.3. The creation of base in-
stances for our numerical experiments is presented in Section 4.4. A heuristic preprocessing
which was used to eliminate unattractive gas stations and thus to reduce the problem size
is introduced in Section 4.5. In Section 4.6, the environment and the different parameter
settings for the numerical experiments are described. Section 4.7 gives a detailed example
for the evolvement of the solution and the driver schedule over the several optimization
steps for one of the test instances. An analysis for all test instances is given in Section
4.8. Section 4.9 makes proposals that can be used to develop a heuristic for the combined
problem.

4.2. Mathematical formulation for the combined
problem

In the following, we show how to merge the two MILP models, the model for refueling
decisions as described in the last chapter and the model for scheduling driving times, rest
periods and breaks from Chapter 2 and which modifications have to be made. Figure 4.1
takes up again the chosen graph structure from Section 3.4.

The sequence of locations consists of an origin and a destination, a sequence of customer
locations to be visited in between, and sequences of gas stations for each pair of consecutive
customer locations including origin and destination. The customer locations are illustrated by the factory symbol and gas stations by the corresponding gas station symbol.

To allow for the consideration of fuel consumption that depends on route lengths and also on properties like road types and geographical data, and to enable the possible usage of historical fuel consumption rates for route segments often traveled, we do not consider fixed fuel consumption rates as input parameters but a concrete fuel consumption for each path between locations. Note that only gas stations that lie in a previously specified (linear or real) distance to the route are relevant and are listed with their detour durations ($\Delta_{drTo}^i$ and $\Delta_{drFrom}^i$ for the detour duration to and from the gas station $i$, respectively) and detour consumptions ($\Delta_{consTo}^i$ and $\Delta_{consFrom}^i$ for the detour consumption to and from the gas station $i$, respectively) as potential refueling points.

Driving durations and fuel consumptions between two consecutive locations $i$ and $i+1$ without detours (in case one or both locations are gas stations) are given by $\Delta_{dr}^{(i,i+1)}$ and $\Delta_{cons}^{(i,i+1)}$, respectively.

Similar to Chapter 2, each customer location has at least one time window, i.e. a time interval in which the loading and/or unloading of goods should start. We consider multiple customer time windows, as in reality, a dispatcher often has the possibility to choose among a set of time windows proposed by a customer. The start of the $z$-th time window interval at location $i$ is given by $TW_{iz}^{begin}$, the end by $TW_{iz}^{end}$ ($z = 0, 1, ..., noFW_i - 1$).

In Section 4.2.1, we describe the modifications that have to be made for constraints adopted from the MILP model described in Section 2.5 to be able to integrate the time aspects of refueling decisions. Additionally, parameters, variables and constraints taken from the fixed sequence refueling problem of Section 3.5 to incorporate refueling are listed. A complete list of all parameters and variables of the whole MILP model is given in Appendix A.
4. The combined problem

4.2.1. Modifications in the model to plan driving times, rest periods and breaks

All constraints from Chapter 2 are adopted for the extended vertex set that now additionally consists of refueling vertices. Note that as described in Section 3.4 a consecutive numbering is chosen for all locations, i.e. for origin, destination, customer locations and gas stations, not differentiating between the kind of location. Some modifications are necessary and will be described in the following.

In the MILP model from Chapter 2, the driving durations between two consecutive locations $\Delta_{(i,i+1)}^{drive}$ were constant. As now gas stations are included in the list of locations, the durations contain a variable part if at least one of the locations is a gas station. Depending on whether a gas station $i$ is chosen for refueling ($\alpha_{i}^{refuel} = 1$) or not ($\alpha_{i}^{refuel} = 0$), driving durations for detours have to be added. Note that $\Delta_{i}^{drTo}$ and $\Delta_{i}^{drFrom}$ are set to be zero if location $i$ is not a gas station. In the equality conditions (4.2.1) to be added to the MILP model, the driving duration from gas station $i$ to the point where the route is entered is added if $i$ is a gas station chosen for refueling. The driving duration to gas station $i+1$ is added if $i+1$ is chosen for refueling.

$$\Delta_{(i,i+1)}^{dr} = \Delta_{(i,i+1)}^{drive} + \Delta_{i}^{drFrom} \alpha_{i}^{refuel} + \Delta_{i+1}^{drTo} \alpha_{i+1}^{refuel} \quad \forall \ i \in S^{locations}$$

(4.2.1)

Similar to Section 3.5, a vertex between the origin and the final destination may be a customer location or a gas station. Two different kinds of working activities may take place: working activities associated with loading and/or unloading goods or refueling. Therefore, variables $\Delta_{i}^{work}$ and constraints (4.2.2) are added.

$$\Delta_{i}^{work} = \Delta_{i}^{service} + \Delta_{i}^{refuel} \alpha_{i}^{refuel} \quad \forall \ i \in S^{locations}$$

(4.2.2)

The constants $\Delta_{(i,i+1)}^{drive}$ are substituted by the variables for driving durations $\Delta_{(i,i+1)}^{dr}$ and the constants $\Delta_{i}^{service}$ are substituted by the variables for the duration of working time at locations $\Delta_{i}^{work}$ in all constraints of the MILP model of Section 2.5 where these constants appear. These changes affect the constraints for the driving time left until the next break or daily rest period when entering vertex $i$ (variables $E_{i}^{dt}$), the daily driving time left until the next daily rest period (variables $E_{i}^{ddt}$) and the overall time left until the next daily rest period (variables $E_{i}^{t}$) upon arrival at a location $i$. Here, $\Delta_{(i,i+1)}^{drive}$ has to be replaced by $\Delta_{(i,i+1)}^{dr}$, $\Delta_{i}^{service}$ is replaced by $\Delta_{i}^{work}$ in the constraints for the time left until the next daily rest period when leaving $i$, $L_{i}^{t}$. In the constraints for the begin of service, both, $\Delta_{(i,i+1)}^{drive}$ and $\Delta_{i}^{service}$ are substituted accordingly.
Time windows are only considered for customer locations and the final destination. The constraints which state that exactly one time window has to be chosen for each location are customized to a limited vertex set (see (4.2.3)). The start of loading and/or unloading is restricted by time windows. We decided to not consider time windows at gas stations (noTW\textsubscript{i}, the number of time windows at location \(i\), is equal to 0 for all \(i\in S\text{stations}\)) and therefore, inequalities (4.2.4), that state that loading and/or unloading only can start after the lower bound of the time window interval, can be adopted without modifications.\textsuperscript{66} The variable \(tw_{iz}\) is equal to 1 if time window \(z\) at location \(i\) is chosen and 0 otherwise.

\[
\sum_{z=0}^{noTW_i-1} tw_{iz} = 1 \quad \forall \ i \in S\text{customers} \cup \{r-1\} \quad (4.2.3)
\]

\[
start_i \geq \sum_{z=0}^{noTW_i-1} TW_{iz}^{\text{begin}} tw_{iz} \quad \forall \ i = 1, \ldots, r-1 \quad (4.2.4)
\]

Lateness at location \(i\) is denoted by \(\Delta_{i}^{\text{late}}\). Since no lateness is considered at gas stations, only the vertex set \(S\text{customers} \cup \{r-1\}\) is covered by the modified lateness constraints (4.2.5).

For gas station vertices, lateness is set to be equal to 0 by the new equations (4.2.6).

\[
\Delta_{i}^{\text{late}} \geq start_i - \sum_{z=0}^{noTW_i-1} TW_{iz}^{\text{end}} tw_{iz} \quad \forall \ i \in S\text{customers} \cup \{r-1\} \quad (4.2.5)
\]

\[
\Delta_{i}^{\text{late}} = 0 \quad \forall \ i \in S\text{stations} \quad (4.2.6)
\]

To restrict the solution space, we allow daily rest periods in vertices associated with gas stations only in cases where they are necessary for refueling because the time left until the next daily rest period is exhausted. The auxiliary binary variables \(\lambda_i^7\) are introduced to model the corresponding constraints (see (4.2.7) to (4.2.11)).

\[
900 \lambda_i^7 \geq E_i^t - \bar{\Delta}_{i}^{\text{refuel}} \alpha_i^{\text{refuel}} \quad \forall \ i \in S\text{stations} \quad (4.2.7)
\]

\[
900 (\lambda_i^7 - 1) \leq E_i^t - \bar{\Delta}_{i}^{\text{refuel}} \alpha_i^{\text{refuel}} \quad \forall \ i \in S\text{stations} \quad (4.2.8)
\]

\[
\alpha_i^{\text{rest}} \leq 1 - \lambda_i^7 \quad \forall \ i \in S\text{stations} \quad (4.2.9)
\]

\textsuperscript{66} Although time windows can be used to model opening hours of gas stations, this adds more complexity to the model and does only make sense if opening hours of gas stations are maintained.
Waiting time, breaks and partial breaks at gas stations are prohibited by constraints (4.2.7), (4.2.8) and (4.2.10) as such activities from a mathematical point of view do not bring any benefits. Waiting time can always be postponed to the next customer location or be used to extend the duration of a daily rest period without worsening the solution value. If a break is needed to reset the time interval until the next break, it can also be taken later on the way from the gas station to the next location, and therefore may be mapped onto the corresponding arc. As there are no additional waiting times considered at gas stations that can be compensated by breaks or partial breaks, \( \alpha_{i}^{\text{break}} \), the variable that is equal to 1 if a break is taken in vertex \( i \) and 0 otherwise, and \( \alpha_{i}^{\text{partial break}} \), the variable that indicates if a partial break is taken in vertex \( i \), are set to be zero for all gas stations \( i \in S_{\text{stations}} \).  

Constraints (4.2.7) set \( \lambda_{i}^{\text{rest}} \) to be equal to 1 if there is still time left until the next daily rest period after refueling without taking a daily rest period in advance. Constraints (4.2.8) ensure that \( \lambda_{i}^{\text{rest}} \) is equal to 0 in case refueling takes place at gas station \( i \) and this is not possible without taking a daily rest period. Constraints (4.2.9) then state that a daily rest period in vertex \( i \) may only be taken if \( \lambda_{i}^{\text{rest}} \) is equal to 0. In case the time needed for refueling suffices exactly without taking a daily rest period, \( \lambda_{i}^{\text{rest}} \) may take on both values, 1 or 0. To ensure that no daily rest period is taken in that case, constraints (4.2.10) are introduced with \( M \) chosen sufficiently large.  

\[
\lambda_{i}^{\text{rest}} \leq M \left( \Delta_{i}^{\text{refuel}} \alpha_{i}^{\text{refuel}} - E_{i} \right) + (M + 1) \cdot 900 \lambda_{i}^{\text{work}} \quad \forall i \in S_{\text{stations}} \tag{4.2.10}
\]

\[
\lambda_{i}^{\text{rest}} = 0 \quad \forall i \in S_{\text{customers}} \cup \{0, r - 1\} \tag{4.2.11}
\]
4.2. Mathematical formulation for the combined problem

\[ \Delta_{\text{wait}}^i = 0 \quad \forall \ i \in S^{\text{stations}} \]  
\[ \alpha_{\text{break}}^i = 0 \quad \forall \ i \in S^{\text{stations}} \]  
\[ \alpha_{\text{pbreak}}^i = 0 \quad \forall \ i \in S^{\text{stations}} \]  

A partial daily rest period \( \alpha_{\text{prest}}^i \) for \( i \in S^{\text{stations}} \) is only allowed if it substitutes a break on the preceding arc (see (4.2.15)). Note that if the variable \( \mu_{\text{prest}}^i \) is equal to one, this indicates that the partial daily rest period planned \( \alpha_{\text{prest}}^i \) is not taken upon arrival at gas station \( i \) but "substitutes" a break between location \( i - 1 \) and gas station \( i \). That means that the last resting activity between location \( i - 1 \) and gas station location \( i \) is a partial daily rest period although the number of breaks on the arc \( A_{(i-1,i)}^{\text{break}} \) would suggest another break.

\[ \alpha_{\text{prest}}^i \leq \mu_{\text{prest}}^i \quad \forall \ i \in S^{\text{stations}} \]  

The refueling constraints (3.5.2) to (3.5.9) and the corresponding variables from the sequence refueling problem formulated in Section 3.5 are adopted.

4.2.2. Objective functions

The most important objective is the minimization of lateness. To keep the overall schedule duration low, i.e. to arrive at the final destination as soon as possible, is the second objective. The third objective is to minimize the overall costs for refueling. Other criteria that are relevant in practice and important for the acceptance by drivers and dispatchers are included in one objective function. Similar as in Section 2.5, a combination of strategies was chosen when setting up the objective functions and determining the solution process for this multicriteria optimization problem. The different objective functions are described in the following. The solution methodology to solve the MILP model is described in Section 4.3.

Objective function 1

For the first objective function, the trade-off strategy from Section 2.5 was chosen, giving most importance to the minimization of lateness. For the choice of the penalty factor \( P \) see Section 2.5.19 (page 83).
4. The combined problem

Minimize \[ \text{start}_{r-1} + \sum_{i=1}^{r-1} P \cdot \Delta_{i}^{\text{late}} \] \hfill (4.2.16)

Objective function 2

The second objective function minimizes the overall refueling costs, where \( \bar{P}_{i} \) denotes the fuel price at gas station \( i \).

Minimize \[ \sum_{i \in \text{stations}} \bar{P}_{i} \cdot \Delta_{i}^{\text{refuel}} \] \hfill (4.2.17)

Objective function 3

Objective function 3 is an extension of objective function 2 described in Section 2.5.19. Note that the last two components are added to the original objective function penalizing the number of refueling stops and the driving duration for the complete route. As the variable part of the route are the detours, the last component penalizes durations for detours. The different weights may be customized depending on user preferences.

\begin{align*}
\text{Minimize} & \sum_{i=1}^{r-1} \sum_{z=0}^{\text{noTW}-1} 10 \left( z + r - i \right) t_{w_{iz}} + \sum_{i=0}^{r-1} \text{start}_{i} \\
& + \sum_{i=0}^{r-2} 10 \left( r - i \right) \left( \mu_{(i,i+1)}^{\text{earlydr}1} + \mu_{(i,i+1)}^{\text{earlydr}2} \right) \\
& + \sum_{i=0}^{r-1} 10 \left( r - i \right) \left( \alpha_{i}^{\text{break}} + \alpha_{i}^{\text{pbreak}} \right) \\
& + \sum_{i=0}^{r-1} 20 \Delta_{i}^{\text{wait}} \\
& + \sum_{i=0}^{r-2} 30 \left( r - i \right) \mu_{(i,i+1)}^{\text{redrest}} + \sum_{i=0}^{r-1} 40 \left( r - i \right) \mu_{i}^{\text{redrest}} \\
& + \sum_{i=0}^{r-2} 50 \left( r - i \right) \mu_{(i,i+1)}^{\text{extd}2} + 60 \left( r - i \right) \mu_{(i,i+1)}^{\text{extd}1} + 60 \left( r - i \right) \mu_{(i,i+1)}^{\text{extd}3}
\end{align*}

\(^{69}\) Note that refueling quantities are set to be zero for non gas station locations by (3.5.6) and (3.5.8).
4.3. The solution process

Since a lexicographical ordering\(^{70}\) of the different objective functions will be used, multiple optimization steps are necessary to solve the multicriteria optimization problem. In each optimization step, a submodel is solved which consists of the constraints described in the previous section and in Section 2.5, and a corresponding objective function. From step 2 onwards additional constraints need to be added. Each step is described in the following.

Punctuality at customer locations is often more important than saving fuel costs, as customer satisfaction has a big impact on future requests and thus on the economic viability of a haulage company. We therefore may order objective functions 1 and 2 lexicographically giving highest importance to objective function 1 (4.2.16).

When setting up the solution process for the MILP model without consideration of refueling, we noticed that it was beneficial to have an additional submodel in which optional rules were deactivated and to use the optimal objective function value of this submodel as an upper cutoff for the submodel in which optional rules were allowed. Using this experience, we adopted the same approach and obtained two submodels and two optimization steps for the first objective function. For details see Chapter 2.

For the objective of minimizing fuel costs, we set up a third submodel with objective function (4.2.17). Two additional constraints are added to this submodel. The first one, (4.3.1), does not allow more lateness than the total lateness over all locations $i$ obtained in optimization step 2.

$$\sum_{i=1}^{r-1} \Delta_{i}^{late} \leq \sum_{i=1}^{r-1} \Delta_{i}^{late^a}$$

(4.3.1)

Note that a solution still has to exist if there was one in the previous steps as refueling already was considered even though not in an optimal way.

---

\(^{70}\) Note that in step 3, the lexicographic ordering is softened to obtain more freedom for refueling.
For more freedom, in optimization step 3 we allow the completion time \( start_{r-1} \) to be at most 30 minutes more than in optimization step 2. This is expressed in constraint (4.3.2), where \( start_{r-1}^* \) denotes the completion time of step 2. The time may be used for an additional and/or alternative refueling.

\[
start_{r-1} \leq start_{r-1}^* + 30
\]  

(4.3.2)

Similarly to the solution process for the model without refueling, an additional submodel and a corresponding optimization step was added to obtain more comprehensible solutions, to only use optional rules if this is advantageous, and to keep the number of refueling stops and detour durations low.

For the additional optimization step, the objective function of the previous step is transformed into constraint (4.3.3) with the optimal objective function value \( z^* \) of step 3 as an upper bound such that the fuel costs are prevented from increasing.

\[
\sum_{i \in S_{stations}} \tilde{P}_i \cdot \Delta_i^{refuel} \leq z^*
\]  

(4.3.3)

Again, constraint (4.3.1) was added to keep the optimal lateness determined in step two.

In step 3, we allow for more freedom for refueling decisions when adding constraint (4.3.2). In step 4, we do not allow an increase of the schedule duration and thus add constraint (4.3.4), where \( start_{r-1}^* \) in that case represents the completion time determined in optimization step 3.

\[
start_{r-1} \leq start_{r-1}^*
\]  

(4.3.4)

The objective function of optimization step 4 is given by (4.2.18).

The solution of optimization step 4 still needs to be transformed into a driver schedule. In Section 2.6, a transformation algorithm was developed for this task. The time for loading and/or unloading goods at customer location \( i \) that was taken from the input parameters of the MILP model has to be replaced by the value of the variable for general working time, \( \Delta_{i}^{work} \), for each customer location or gas station \( i \). Similarly, the driving time between a pair of consecutive locations \( i \) and \( i+1 \) is now variable and given by \( \Delta_{(i,i+1)}^{dr} \). This has to be adopted for the input parameters of the algorithm accordingly.

Figure 4.2 gives an overview of the solution process.
In the next section, the test instances are presented. Afterward, in Section 4.5, a preprocessing heuristic is introduced which helps to reduce the number of gas stations to be considered during the solution process. In our numerical experiments, all subproblems are solved with a commercial optimization solver. The details on the test environment are described in Section 4.6. In Section 4.7 it is shown for a test instance how the driver schedule evolves over the several optimization steps.

4.4. Numerical experiments - The base instances

We extended the data basis of the test instances that were used to test the MILP model for planning time windows, driving times, rest periods and breaks (see Section 2.7.1) by adding information about gas stations along the route. In a first step, we decided to consider gas stations with a straight line (i.e. Euclidean) distance of at most 30 km to the route.

To obtain driving durations and distances between locations and for detours, a modified A* routing algorithm was used which was developed during the research project Dynaserv. The real vehicle fuel consumptions for the one-week routes were used to determine distance dependent consumption rates. Real data on the tank capacities of the vehicles, start
and end fuel levels at the beginning and the end of the planning horizon were adopted. For simplification reasons, detour durations and consumptions from gas stations back to the route were assumed to be equivalent to the values obtained for the detours to the corresponding gas stations. For the sorting of gas stations between a pair of consecutive customer locations the point where the route has to be left to head for the corresponding gas station was the decisive criterion.

List prices per country of one of the main fuel card operators of the haulage company were provided. The fuel card operator considers different types of gas stations, among those the group of gas stations that are close to the highway and therefore are more expensive and a group of gas stations that are less expensive. In reality, the fuel price at a gas station that is valid for the corresponding fuel card holder is dependent on the gas station type and on the list price of the corresponding country. Additionally, there are special discount arrangements for selected gas stations. For Spain, there was a contract with a different service station chain. For simplification reasons, fuel prices at gas stations were assumed to be equal to the list prices of the corresponding countries at the beginning of the corresponding planning horizons.

Table 4.1 gives an overview of the extended base instances.

<table>
<thead>
<tr>
<th>base instance</th>
<th>overall distance (km)</th>
<th>overall driving duration (h)</th>
<th># customer locations (incl. start &amp; end)</th>
<th>tank capacity (l)</th>
<th>start fuel level (l)</th>
<th>end fuel level (l)</th>
<th>fuel consumption (l)</th>
<th># gas stations (30 km straight line distance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2914</td>
<td>36.85</td>
<td>4</td>
<td>925.00</td>
<td>925.00</td>
<td>619.75</td>
<td>670.30</td>
<td>276</td>
</tr>
<tr>
<td>2</td>
<td>3391</td>
<td>42.48</td>
<td>5</td>
<td>925.00</td>
<td>583.75</td>
<td>753.50</td>
<td>803.59</td>
<td>397</td>
</tr>
<tr>
<td>3</td>
<td>3653</td>
<td>46.95</td>
<td>6</td>
<td>900.00</td>
<td>360.00</td>
<td>837.00</td>
<td>1077.72</td>
<td>181</td>
</tr>
<tr>
<td>4</td>
<td>2851</td>
<td>36.45</td>
<td>6</td>
<td>900.00</td>
<td>873.00</td>
<td>468.00</td>
<td>781.30</td>
<td>447</td>
</tr>
<tr>
<td>5</td>
<td>1739</td>
<td>22.37</td>
<td>6</td>
<td>925.00</td>
<td>268.25</td>
<td>900.00</td>
<td>486.82</td>
<td>241</td>
</tr>
<tr>
<td>6</td>
<td>2944</td>
<td>37.47</td>
<td>6</td>
<td>900.00</td>
<td>729.00</td>
<td>396.00</td>
<td>889.12</td>
<td>326</td>
</tr>
<tr>
<td>7</td>
<td>2269</td>
<td>30.32</td>
<td>7</td>
<td>900.00</td>
<td>495.00</td>
<td>639.00</td>
<td>664.95</td>
<td>284</td>
</tr>
<tr>
<td>8</td>
<td>3142</td>
<td>39.77</td>
<td>7</td>
<td>900.00</td>
<td>666.00</td>
<td>891.00</td>
<td>816.86</td>
<td>374</td>
</tr>
<tr>
<td>9</td>
<td>3019</td>
<td>38.17</td>
<td>7</td>
<td>925.00</td>
<td>712.25</td>
<td>910.00</td>
<td>830.21</td>
<td>515</td>
</tr>
<tr>
<td>10</td>
<td>3436</td>
<td>43.77</td>
<td>8</td>
<td>900.00</td>
<td>747.00</td>
<td>693.00</td>
<td>896.92</td>
<td>383</td>
</tr>
<tr>
<td>11</td>
<td>3447</td>
<td>43.62</td>
<td>8</td>
<td>1200.00</td>
<td>504.00</td>
<td>444.00</td>
<td>1082.25</td>
<td>474</td>
</tr>
<tr>
<td>12</td>
<td>2475</td>
<td>31.85</td>
<td>9</td>
<td>900.00</td>
<td>873.00</td>
<td>684.00</td>
<td>737.67</td>
<td>298</td>
</tr>
<tr>
<td>13</td>
<td>2826</td>
<td>36.42</td>
<td>10</td>
<td>900.00</td>
<td>648.00</td>
<td>576.00</td>
<td>802.50</td>
<td>337</td>
</tr>
<tr>
<td>14</td>
<td>3055</td>
<td>40.85</td>
<td>11</td>
<td>900.00</td>
<td>666.00</td>
<td>576.00</td>
<td>837.06</td>
<td>368</td>
</tr>
<tr>
<td>15</td>
<td>3250</td>
<td>41.95</td>
<td>12</td>
<td>900.00</td>
<td>801.00</td>
<td>360.00</td>
<td>952.28</td>
<td>353</td>
</tr>
</tbody>
</table>

Table 4.1.: Base instances

In the first column, the ID of each of the 15 base instances is considered. Distances and driving durations without consideration of detours are given in the second and third column, respectively. The number of customer locations including the origin and destination are given in the fourth column. The vehicles considered have different tank capacities that range from 900 to 1200 liters. The tank capacities are given in the fifth column. Start and end fuel levels are given in columns six and seven. Column eight shows the overall fuel consumption for the one-week routes. This fuel consumption divided by the overall distance gave us the fuel consumption rate in liters per km. The number of gas stations within a straight line distance of at most 30 km along the route is shown in the last column. Note that gas stations were chosen per route between consecutive locations in $S_{customers} \cup \{0, r - 1\}$. Gas stations that were within the chosen straight line distance for several of such routes were listed multiple times accordingly.
4.5. Preprocessing Heuristic: Eliminating unattractive gas stations

As we will see later in our numerical experiments, the number of gas stations included in the list of potential gas stations for refueling strongly influences the time needed by the optimization solver to find a solution. Therefore, a preprocessing heuristic was developed to eliminate less promising gas stations from the list and thus reduce the computational efforts necessary in the following steps.

As mentioned earlier in Section 3.3, Suzuki (2014) also proposes a preprocessing procedure that reduces the number of gas stations to be considered. Note that this procedure applied to our problem may remove attractive gas stations as we also consider time factors which have a high priority in our problem definition. The time needed for a detour is not considered in the elimination process described by Suzuki (2014). Conversely, one criterion for the elimination of a gas station is that its detour distance is less than the detour distance of two other gas stations that represent the start and the end point of a subsequence of gas stations. Eliminating such gas stations with short detour distances can be disadvantageous if customer time windows are involved. Additionally, the average number of gas stations removed by the variable-reduction technique does not sufficiently reduce the problem size in preparation for the solution process for the MILP model provided in this study.\(^\text{71}\)

The heuristic presented in this section is applied to each pair of consecutive customer locations (including origin and final destination) and has a run time complexity of \(O(n^2)\), where \(n\) is the number of gas stations between the two locations. For each route between consecutive customer locations (or between origin and first customer location or last customer location and final destination), gas stations within the chosen straight line distance of 30 km to the route were sorted ascending by price and by detour distance, where the price was chosen to be the first sorting criterion. To obtain the sorted list of gas stations, we used the sorting algorithm of the Collections package of Java (java.util.Collections) together with a comparison function (using java.util.Comparator). For two gas stations \(i\) and \(j\), the comparison function returns \(-1\) if gas station \(i\) according to the sorting criteria has to stand higher in the list than gas station \(j\), \(1\) if \(j\) has to stand higher in the list than \(i\) and \(0\) otherwise. Nevertheless, the sorting algorithm can be chosen independently. As sorting algorithms are broadly discussed in the literature, we only present the comparison function, Algorithm 8, used for the sorting in Java. Note that in the following we assume that the detour to a gas station is equal to the detour back to the route. Additionally, linear fuel consumption rates per distance unit are assumed.

\(^{71}\) For the instances considered by Suzuki (2014), the number of gas stations was reduced by 54.8\% on average.
Algorithm 8 Compare price and detour

1: \textbf{compare}(\bar{P}_i, \bar{P}_j, \text{detour}_i, \text{detour}_j)

\textbf{Input:}

\begin{align*}
&\bar{P}_i (\bar{P}_j) : \quad \text{Fuel price at gas station } i (j) \\
&\text{detour}_i (\text{detour}_j) : \quad \text{Detour distance if gas station } i (j) \text{ is visited}
\end{align*}

\textbf{Output:}

\begin{align*}
\text{return } & \begin{cases} 
-1 & \text{if gas station } i \text{ should stand higher in the list than gas station } j, \\
1 & \text{if gas station } j \text{ should stand higher in the list than gas station } i, \\
0 & \text{otherwise}
\end{cases}
\end{align*}

2: // First sorting criterion: Fuel price (ascending)
3: if \( \bar{P}_i < \bar{P}_j \) then
4: return \(-1\)
5: else if \( \bar{P}_i > \bar{P}_j \) then
6: return \(1\)
7: else
8: // Second sorting criterion: Detour distance (ascending)
9: if \( \text{detour}_i < \text{detour}_j \) then
10: return \(-1\)
11: else if \( \text{detour}_j < \text{detour}_i \) then
12: return \(1\)
13: else
14: return \(0\)
15: end if
16: end if

The following comparison function (Algorithm 9) is necessary to sort the gas stations by the sequence in which the points where the route has to be left to reach the gas stations are traversed. The corresponding sorting is done as a preprocessing step to determine input parameters for the MILP model. In the MILP model, the driving durations and fuel consumptions between consecutive locations are needed. If \( i \) and \( j \) are two consecutive gas stations, the distance between them can for example be determined by subtracting the distance on the route between the last customer location and gas station \( i \) (\( \text{dist}_i \)) from the distance on the route between the last customer location and gas station \( j \) (\( \text{dist}_j \)). For the driving durations and fuel consumptions this can be done analogously.
4.5. Preprocessing Heuristic: Eliminating unattractive gas stations

Algorithm 9 Compare on-route distance

1: \texttt{compare}(\texttt{dist}_i, \texttt{dist}_j)

\textbf{Input:}

\begin{itemize}
  \item \texttt{dist}_i \ (\texttt{dist}_j) : \text{Distance on the route between the last customer location and gas station } i \ (j)
\end{itemize}

\textbf{Output:}

\begin{itemize}
  \item return \{-1 if gas station } i \ \text{should stand higher in the list than gas station } j, \ 1 \ \text{otherwise}
\end{itemize}

2: if \texttt{dist}_i < \texttt{dist}_j then
3: return -1
4: else
5: return 1
6: end if

We want to keep the "best" gas stations considering the two criteria, fuel price and detour distance. Those gas stations for which there is a "better" gas station considering both criteria, fuel price and detour distance, in a predefined on-route distance are eliminated. In the following, we call this predefined distance "filter distance". Algorithm 10 shows the elimination process. Note that if gas station } j \ \text{stands higher in the list than gas station } i \ \text{and the list has been sorted using the comparison function given in Algorithm 8, } j \ \text{has definitely a fuel price that is lower than or equal to the fuel price of } i. \ \text{We therefore only compare the detour distances of gas stations } i \ \text{and } j \ \text{in Algorithm 10.}

Algorithm 10 Filter gas stations

\textbf{Input:}

\begin{itemize}
  \item \texttt{P}_i(\texttt{P}_j) \ \text{Fuel price at gas station } i \ (j)
  \item \texttt{detour}_i(\texttt{detour}_j) \ \text{Detour distance (one-way) if gas station } i \ (j) \ \text{is visited}
  \item \texttt{dist}_i \ \text{Distance on the route from the last customer vertex (or origin) to the point where the route has to be left if gas station } i \ \text{should be visited}
  \item n \ \text{Number of gas stations between the two customer locations considered}
  \item \texttt{S}_{\text{stations}} \ (c,c+1) \ \text{Set of all gas stations } i \ \text{with selected straight line distance to the route between customer } c \ \text{and customer } c + 1.
  \item radius \ \text{Filter distance}
\end{itemize}
Output:

\[ \hat{S}_{\text{stations}}^{(c,c+1)} \]

Remaining list of gas stations between customer locations \( c \) and \( c + 1 \)

1: Sort the gas stations in \( S_{\text{stations}}^{(c,c+1)} \) by price (first criterion) and by detour distance (second criterion) using Algorithm 8. Resulting list: \( \hat{S}_{\text{stations}}^{(c,c+1)} \)

4: // Go through the sorted list of gas stations, start by the second station.
5: // (The first station is kept in the list as it has the lowest price).

7: for \( i = 1 \) to \( n - 1 \) do

8: // Go through the list of gas stations that were kept and only keep gas station \( i \)
9: // in the list if for one of the two criteria (this can only be the detour distance)
10: // it is better than all gas stations kept so far or no kept gas station lies in the
11: // filter distance of gas station \( i \).

14: for \( j = 0 \) to \( i - 1 \) do

15: if \( \text{detour}_i \geq \text{detour}_j \) then
16: if \( |\text{dist}_i - \text{dist}_j| < \text{radius} \) then
17: \( \hat{S}_{\text{stations}}^{(c,c+1)} \leftarrow \hat{S}_{\text{stations}}^{(c,c+1)} \setminus \{i\} \)
18: \( i = i - 1 \)
19: break
20: end if
21: end if
22: \( j = j + 1 \)
23: end for
24: \( i = i + 1 \)
25: end for

37: Sort the gas station list \( \hat{S}_{\text{stations}}^{(c,c+1)} \) by the distance on the route between the last customer location \( c \) and the gas station using Algorithm 9. Result: Newly sorted list \( \hat{S}_{\text{stations}}^{(c,c+1)} \).
39: return \( \hat{S}_{\text{stations}}^{(c,c+1)} \)
Figure 4.3 illustrates Algorithm 10 by means of an example.

![Diagram of Eliminating unattractive gas stations]

**Figure 4.3:** Example of filtering gas stations

The numbering of gas stations depicted is the numbering obtained by sorting the gas stations with Algorithm 8. The sequence in which the gas stations are considered is equal to this numbering. The gas station with number 1 is kept, as it is the cheapest (and it is the first one in the list) among all gas stations between customer \( c \) and customer \( c + 1 \). The second gas station is kept because the detour distance is less than the detour distance of gas station 1. Gas station 3 is eliminated because it is within the filter distance of gas station 1 (and 2) and it has a higher fuel price and a larger detour distance than gas station 1. Gas station 4 is within the filter distance of gas station 1, but it has a smaller detour distance, so it is kept. Gas station 5 is removed as it is within the filter distance of gas station 4 and has a larger detour distance (and equal fuel price). Gas station 6 is outside the filter distance of any of the gas stations already considered and thus is kept in the list. Gas station 7 has a shorter detour distance than gas station 6, which is the only remaining gas station with lower index within the filter distance of gas station 7. Thus, it is not eliminated. Gas station 8 is within the filter distance of gas stations 4, 6 and 7 and is removed because it has a higher fuel price and a larger detour distance than gas station 7. From the initially considered 8 gas stations, three are eliminated.

Note that for the choice of the filter distance the tank capacity has to be taken into account. If the filter distance is larger than one half of the distance that can be traveled with a full tank, no solution might be found for the corresponding MILP model. The maximum detour distance may also have an influence when it is large. Additionally, the start and end fuel levels have to be taken into account. Therefore, for the filtering of gas stations between the origin and the first customer and between the last customer and the final destination, the filtering algorithm is slightly modified. To ensure that the first gas station on the route is reachable, we use Algorithm 11. At least one gas station is kept by the "normal" filtering algorithm. If the range with the start fuel level is at least as far as the route length between the origin and the first customer plus the maximum detour distance to a gas station, any gas station chosen on this arc is reachable. The original algorithm,
Algorithm 10, is executed. Otherwise, in each iteration, we only remove a gas station from
the list if a first gas station was found that is reachable with the start fuel level.

Algorithm 11 Filter gas stations between origin and first customer location

**Input:**

- $\bar{P}_i$ ($\bar{P}_j$) Fuel price at gas station $i$ ($j$)
- $\text{detour}_i$ ($\text{detour}_j$) Detour distance (one-way) if gas station $i$ ($j$) is visited
- $\text{dist}_i$ Distance on the route from the last customer vertex (or origin) to the point where the route has to be left if gas station $i$ should be visited
- $n$ Number of gas stations between origin and first customer location
- $\mathcal{S}_{\text{stations}}^{(0,1)}$ Set of all gas stations $i$ with selected straight line distance to the route between origin $0$ and first customer $1$.
- $\text{radius}$ Filter distance
- $\text{rangeStartFuel}$ Maximum distance traveled by the vehicle with the start fuel level
- $\text{routeLength}$ Length of the route between origin and first customer
- $\text{maxDetour}$ Maximum detour distance to a gas station (one-way)

**Output:**

$\hat{\mathcal{S}}_{\text{stations}}^{(0,1)}$ Remaining list of gas stations between origin $0$ and customer location $1$

1: // Check if the execution of algorithm 10 suffices.
2: if $(\text{routeLength } + \text{maxDetour} \leq \text{rangeStartFuel})$ then
3: return $\hat{\mathcal{S}}_{\text{stations}}^{(0,1)}$ obtained by Algorithm 10
4: else
5: Sort the gas stations in $\mathcal{S}_{\text{stations}}^{(0,1)}$ by price (first criterion) and by detour distance (second criterion) using Algorithm 8. Resulting list: $\hat{\mathcal{S}}_{\text{stations}}^{(0,1)}$
6: $\text{gasStationInRange} = \text{false}$
7: // Go through the sorted list of gas stations, start by the second station
8: // (the first station is kept).
4.5. Preprocessing Heuristic: Eliminating unattractive gas stations

for $i = 1$ to $n - 1$ do
  for $j = 0$ to $i - 1$ do
    if not $\text{gasStationInRange}$ then
      if $(\text{dist}_j + \text{detour}_j) \leq \text{rangeStartFuel}$ then
        $\text{gasStationInRange} \leftarrow \text{true}$
      end if
    end if
    if $\text{detour}_i \geq \text{detour}_j$ then
      if $|\text{dist}_i - \text{dist}_j| < \text{radius}$ then
        if $\text{gasStationInRange}$ then
          $\hat{S}^{\text{stations}}_{(0,1)} \leftarrow \hat{S}^{\text{stations}}_{(0,1)} \setminus \{i\}$
          $i = i - 1$
        end if
      end if
      $j = j + 1$
    end if
  end for
  $i = i + 1$
end for

Sort the gas station list $\hat{S}^{\text{stations}}_{(0,1)}$ by the distance on the route between the origin 0 and the gas station using Algorithm 9. Result: newly sorted list $\hat{S}^{\text{stations}}_{(0,1)}$.

return $\hat{S}^{\text{stations}}_{(0,1)}$

end if // ELSE (not $\text{routeLength} + \text{maxDetour} \leq \text{rangeStartFuel}$)
For the filtering of gas stations between the last customer and the final destination, we have to make sure that an arrival with the given end fuel level is possible. This can be done similarly as in Algorithm 11 with the difference that gas stations are only removed from the list if a gas station has been found with a distance to the final destination that is less than the range with a full tank minus the range with the end fuel level. We therefore replace line 23 by an if-statement that checks if the distance of gas station \( j \) to the destination is less than or equal to the range with a full tank minus the range with the end fuel level (\( \text{routeLength} - \text{dist}_j + \text{detour}_j \leq \text{range} - \text{rangeEndFuel} \), \( \text{rangeEndFuel} \): range of the vehicle with the end fuel level, \( \text{routeLength} \): Length of the route between the last customer and the final destination). The if-statement in line 3 is modified with the inequality \( \text{routeLength} + \text{maxDetour} \leq \text{range} - \text{rangeEndFuel} \).

In the tests described in the following sections, different filter distances were considered. The real maximum average fuel consumption over all base instances was less than 321 per 100 km distance traveled. The minimum tank capacity was 900 l and we decided for a minimum fuel level which had to be maintained in the tank at all times to not run out of fuel in case of unforeseen events (e.g. traffic jam) of 100 l. With this information, we computed a minimum range with a full tank of \( 900 l - 100 l = 2500 \) km. We decided to consider a filter distance of at most 1000 km such that, provided that detour distances to gas stations are not "too large" \(^72\), the filtering most likely does not lead to infeasibility of the MILP model set up later. Infeasibility after filtering may occur if even when considering all gas stations along the route there is no possibility to find a choice of gas stations where the minimum fuel level in the tank can be maintained. It also theoretically may happen that the maximum weekly driving time or the maximum time between two weekly rest periods is exceeded because of unfavorable positions of the remaining gas stations as far as the resulting time schedule is considered. But this may not be predicted easily. In all of our test instances, feasibility was preserved for all of the filter distances used.

Table 4.2 shows the remaining number of gas stations (left-hand side) and the overall number of locations (right-hand side) depending on the filter distance used. Note that the number of gas stations per pair of consecutive customer locations (including start and end) were added up for the complete route. Some of the gas stations may occur more than once between different customer locations.

On average, 350 gas stations were found within a straight line distance of 30 km to the route of which 24 (7%) remained when considering a filter distance of 100 km. With a filter distance of 200 km, only 15 (4%) gas stations remained, with 300 km, 13 (4%) gas stations. The number of gas stations in none of the base instances differed by more than 2 when using the filter distances 400 km and 500 km, respectively, and the average number of filtered gas stations was about 11 (3%) in both cases. Finally, a filter distance of 1000 km was considered, leaving on average 9 (3%) gas stations for the optimization process.

\(^72\) In the worst case, two remaining consecutive gas stations (with a customer in between) after filtering may have an on-route distance of at most 2000 km even though before filtering there were gas stations "in between". In such a case where the distance between two consecutive gas stations exceeds 1000 km, the gas stations belong to two consecutive route segments, where a route segment is defined to be the route between two customer locations including origin and destination. Thus, for the detour distance from the last gas station to the subsequent gas station, at least \( (2500 - 2000) \frac{km}{1000} = 500 km \) are remaining from the range with a full tank for the detour from the first of the two gas stations to the route and the detour to the second one. In none of the considered cases, one-way detour distances were larger than 192 km. But if different input parameters are chosen or different properties are observed, this may have to be taken into account.
4.6. Numerical experiments - Environment and settings

<table>
<thead>
<tr>
<th>Base instance</th>
<th># customer locations incl. start and end</th>
<th>no price and detour filter</th>
<th>no price and detour filter</th>
<th>100 km</th>
<th>200 km</th>
<th>300 km</th>
<th>400 km</th>
<th>500 km</th>
<th>1000 km</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>276</td>
<td>22</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>280</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>397</td>
<td>24</td>
<td>13</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>402</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>381</td>
<td>26</td>
<td>15</td>
<td>11</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>487</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>447</td>
<td>23</td>
<td>14</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>453</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>241</td>
<td>15</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>247</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>376</td>
<td>24</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>332</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>284</td>
<td>22</td>
<td>14</td>
<td>12</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>291</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>374</td>
<td>32</td>
<td>19</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>381</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>515</td>
<td>25</td>
<td>15</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>8</td>
<td>522</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>383</td>
<td>25</td>
<td>19</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>391</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>474</td>
<td>27</td>
<td>18</td>
<td>13</td>
<td>11</td>
<td>11</td>
<td>8</td>
<td>482</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>296</td>
<td>23</td>
<td>16</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>307</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>337</td>
<td>20</td>
<td>13</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>347</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>368</td>
<td>27</td>
<td>18</td>
<td>16</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>379</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>352</td>
<td>30</td>
<td>22</td>
<td>21</td>
<td>19</td>
<td>17</td>
<td>15</td>
<td>365</td>
</tr>
</tbody>
</table>

Table 4.2.: Remaining locations after filtering

4.6. Numerical experiments - Environment and settings

For the following numerical experiments, the same test environment was used as for testing the MILP models and the myopic heuristic in Sections 2.7 and 2.9. That means, the MILP model was implemented in Java (Java 8, 64 bit) and solved with CPLEX 12.6 (64 bit) with the ILOG CPLEX Concert Technology. The test runs were performed on an Intel Core i5 2500K with 8 GB RAM (DDR3-10700 (667 MHz)) running Windows 7 Professional Service Pack 1, 64 bit.

Table 4.3 shows the average number of variables and constraints over all test instances with one time window per customer location and depending on the filter distance chosen. In the last line, the values for the MILP model of Chapter 2 are listed for comparison. The number of binary variables in a test instance with more than one time window per customer location is raised by 1 for each additional time window. If the number of time windows at customer locations is constant, the number of additional binary variables is equal to the number of customer vertices, |V_{customers}|, for each additional time window per customer location.

In Section 2.7.1, 225 test instances were generated from 15 base instances (real data). These are now enriched with information needed for refueling (see Section 4.4). For each of the 225 original test instances, 6 different filter distances (100, 200, 300, 400, 500 and 1000 km) were tested. Thus, 1350 complete test runs were performed.

---

73 The number of alternative time windows may differ among customer locations. For each time window z at customer location i, there is a binary decision variable t_{wz,i} in the MILP model (also see Section 4.2.1).
Table 4.3.: Average number of variables and constraints (one time window) depending on the filter distance

While the preprocessing algorithm took less than one second for each of the instances, from previous test runs and from the experience we made in Section 2.7.2, it seemed reasonable to establish maximum run times for the different optimization steps to solve the MILP model. Figure 4.4 shows the allocated time for each step.

Figure 4.4.: Time limits of each optimization step for the MILP model

The idea was to not allow more than half an hour time for the overall solution process unless no solution could be found until then. In none of the optimization steps the solution
process was stopped if no solution was found so far because we wanted to know the run times to find feasible solutions for these cases. As total lateness is considered to be more crucial than total refueling cost and because we knew that the first two steps were the most time consuming, we decided to allow at most 25 minutes for step 1 and 2. If no optimal solution was found in step 1 within 25 minutes, the best solution identified by CPLEX was saved. If no solution at all was found during this time interval, the optimization solver was not stopped until the first solution was found. The same was done in step 2 allowing at most 25 minutes minus the duration of the first step in case at least one feasible solution was found. In step 3, a maximum of 30 minutes minus the durations of steps 1 and 2 were allowed. The remaining time was dedicated to step 4.

4.7. Solution process - An example

In the following, we will show by means of an example how the driver schedule evolves over the several optimization steps. We chose the base instance 3 with three time windows per customer location and a time window length of 30 minutes (see Table 4.4), and considered a filter distance of 1000 km. The same instance (without considering information concerning refueling) was used in Section 2.9.1 to compare different planning techniques. Note that the driver starts his work week at the first customer (marked by a green dot in Figure 4.5), i.e. there is no driving time between start location 0 and the customer location numbered 1. The final destination is chosen to be equal to the final destination reached by the driver in real life and is marked by a red dot in Figure 4.5. The driver was heading to a stop in Wolfsburg when his weekly rest period had to be started.

<table>
<thead>
<tr>
<th>start</th>
<th>Mon 07:47</th>
</tr>
</thead>
<tbody>
<tr>
<td>target location</td>
<td>Rastatt (DE)</td>
</tr>
<tr>
<td>stops</td>
<td>[0]</td>
</tr>
<tr>
<td>duration loading/unloading (h)</td>
<td>2:00</td>
</tr>
<tr>
<td>time windows</td>
<td></td>
</tr>
<tr>
<td>start</td>
<td>Mon 06:30</td>
</tr>
<tr>
<td>end</td>
<td>Mon 07:00</td>
</tr>
<tr>
<td>start</td>
<td>Mon 09:00</td>
</tr>
<tr>
<td>end</td>
<td>Mon 09:30</td>
</tr>
<tr>
<td>start</td>
<td>Mon 11:30</td>
</tr>
<tr>
<td>end</td>
<td>Mon 12:00</td>
</tr>
</tbody>
</table>

Table 4.4.: Time windows

The problem has 6 stops\textsuperscript{74} associated with 4 customer locations, an origin and the final destination, here numbered from 0 to 5. After the execution of the filter algorithm, 7 gas stations located in France, Spain and Belgium were retained. Gas stations in Germany and the Netherlands were in the original list of gas stations along the route but were eliminated.

\textsuperscript{74} Note that even though the driver starts his week with loading and/or unloading at the first customer location, in the MILP model, there is an additional vertex for the start location. This vertex has been added for modeling purposes and for reasons of standardization (e.g. the origin location never has time windows) and thus is also depicted in Table 4.4. The distance between the artificial origin and the first customer location is zero.
because of their high prices. Note that we consider list prices per country which are depicted in Figure 4.5. For more details on the example see Table 4.1 on page 180 for base instance 3. The transformation algorithm was executed after each optimization step to obtain visually comparable results. Note that in all of the steps described in the following, no time limit was reached, i.e. for all of the subproblems an optimal solution was obtained by CPLEX.

Figure 4.5.: Base instance 3

In optimization step 1 (see Table 4.5), the MILP submodel without optional rules was solved, minimizing lateness and completion time. Refueling was already considered but not in an optimal way. With the filtered list of gas stations as possible choices, the driver has to refuel the vehicle in France at the beginning of his trip since with the remaining amount of fuel in the tank he cannot reach a gas station in Spain. He also has a refueling stop in Belgium at the end of his trip as otherwise it is not possible to obtain the high end fuel level of 8371 at the final destination. An additional refueling stop was chosen to

---

75 For the representation of the schedule as depicted in Tables 4.5 to 4.8, the mapping of customer locations to stops as described in Section 4.2 on page 152 in the MILP model is used.
take place in Spain. The fuel costs amount to a total of 1893 € for the whole trip where the final destination is reached with 401 more in the tank than actually needed for the minimum end fuel level.

The optimal objective function value is chosen as an upper cutoff for the MILP submodel in step 2 (see Table 4.6), where the optional rules are considered. If we compare the two schedules, we can see that the overall lateness in step 2 is reduced by the use of the optional rules by 1:27 h (44%). Two reduced daily rest periods between customer stop 2 and 3 allow a punctual arrival at an earlier customer time window. The choice of an earlier time window without causing lateness is also possible for customer stop 4. With the help of a splitted break with one part before loading and/or unloading at customer location 4 and the other part on the way to customer location 5, two extended daily driving times and another reduced daily rest period, the driver arrives at the final destination at 13:12 on Friday and not at 3:27 on Saturday. That are 14:15 h earlier. The choice of gas stations did not change between steps 1 and 2, only the refueling quantities. This time, the final destination is reached exactly with the pre-specified minimum end fuel level. The overall fuel costs are 1857 €.

In step 3 (see Table 4.7), the fuel costs have been optimized, while the lateness from the previous step was prevented from increasing. The completion time was allowed to increase by a maximum of half an hour. In the schedule, the second daily rest period between customer locations 2 and 3 is turned to a regular one. The replacement of the time window chosen for customer stop 3 by the one chosen in step 1 allows another refueling stop on the way to customer 3 without causing lateness. Thus, refueling can take place at the first reachable gas station in Spain (second refueling stop). As the arrival at the final destination is allowed to be 30 minutes later, there is time for an additional refueling stop in Spain between customer location 4 and the final destination. The chosen gas station is the last one in Spain in the filtered list. The complete refueling plan can be described as follows: the driver refuels in France as the next cheaper gas station in Spain would not be reachable, otherwise. The refueling amount is just enough to reach the first gas station in Spain with the minimum fuel quantity allowed in the tank. He fills up completely as there is no cheaper gas station along the route. The last gas station before leaving Spain is used for an additional refueling stop and again, refueling is done until the tank capacity is reached. The last refueling stop is necessary to comply with the predefined end fuel level. The total fuel cost is 1841 €.

The last optimization step (see Table 4.8) serves as a postprocessing with the purpose to obtain more comprehensible solutions and to allow more freedom for re-planning if necessary or for the continuation after the current planning horizon. Constraints are set up to not worsen lateness, completion time and costs for refueling in this step. Optional rules should only be used if this is advantageous and as late as possible. Waiting time should be reduced to a minimum and arrival times at customer locations should be as early as possible. Additionally, the number of refueling stops and detour durations should be kept low. The last objective function takes into account all these criteria. Thus, the second daily rest period is reduced to the minimum duration of a regular daily rest period and waiting time at customer location 3 is omitted. The daily rest period on the route to customer location 4 is extended by 3 minutes such that loading and/or unloading can start at the lower bound of the chosen time window. The arrival at the final destination is 10 minutes earlier than in step 3. The refueling strategy remains the same.
### Table 4.5: Schedule from optimization step 1

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Day</th>
<th>Type</th>
<th>From</th>
<th>Until</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>chosen TW:</td>
<td>start:</td>
<td>Mon</td>
<td>06:30</td>
<td>end:</td>
<td>Mon</td>
</tr>
<tr>
<td>lateness:</td>
<td>0:47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mon</td>
<td>drive</td>
<td>09:47</td>
<td>12:03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mon</td>
<td>(un)load</td>
<td>12:03</td>
<td>14:03</td>
</tr>
</tbody>
</table>

| stop: 2 | chosen TW: | start: | Mon | 10:30 | end: | Mon | 11:00 |
| lateness: | 1:03 |
| | Mon | drive | 14:03 | 16:17 | 2:14 |
| | Mon | rest | 16:17 | 17:02 | 0:45 |
| | Mon | drive | 17:02 | 17:58 | 0:56 |
| | Mon | refuel | 17:58 | 18:18 | 0:20 |
| | Mon | drive | 18:18 | 20:47 | 2:29 |
| | Mon | rest | 20:47 | 07:47 | 11:00 |
| | Tue | drive | 07:47 | 12:17 | 4:30 |
| | Tue | rest | 12:17 | 13:02 | 0:45 |
| | Tue | drive | 13:02 | 17:32 | 4:30 |
| | Tue | rest | 17:32 | 04:32 | 11:00 |
| | Wed | drive | 04:32 | 09:02 | 4:30 |
| | Wed | rest | 09:02 | 09:47 | 0:45 |
| | Wed | drive | 09:47 | 12:27 | 2:40 |
| | Wed | (un)load | 12:27 | 14:27 | 2:00 |

| stop: 3 | chosen TW: | start: | Wed | 10:30 | end: | Wed | 11:00 |
| lateness: | 1:27 |
| | Wed | drive | 14:27 | 16:17 | 1:50 |
| | Wed | rest | 16:17 | 04:15 | 11:58 |
| | Thu | drive | 04:15 | 05:00 | 0:45 |
| | Thu | refuel | 05:00 | 05:20 | 0:20 |
| | Thu | drive | 05:20 | 06:00 | 0:40 |
| | Thu | (un)load | 06:00 | 06:00 | 2:00 |

| stop: 4 | chosen TW: | start: | Thu | 06:00 | end: | Thu | 06:30 |
| lateness: | 0:00 |
| | Thu | drive | 08:00 | 11:05 | 3:05 |
| | Thu | rest | 11:05 | 11:50 | 0:45 |
| | Thu | drive | 11:50 | 16:20 | 4:30 |
| | Thu | rest | 16:20 | 03:20 | 11:00 |
| | Fri | drive | 03:20 | 07:50 | 4:30 |
| | Fri | rest | 07:50 | 08:35 | 0:45 |
| | Fri | drive | 08:35 | 13:05 | 4:30 |
| | Fri | rest | 13:05 | 00:05 | 11:00 |
| | Sat | drive | 00:05 | 02:15 | 2:10 |
| | Sat | refuel | 02:15 | 02:35 | 0:20 |
| | Sat | drive | 02:35 | 03:27 | 0:52 |

| stop: 5 | chosen TW: | start: | Mon | 00:00 | end: | Sun | 23:59 |
| lateness: | 0:00 |
### 4.7. Solution process - An example

<table>
<thead>
<tr>
<th>Step 2</th>
<th>day</th>
<th>type</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mon</td>
<td>(un)</td>
<td>07:47</td>
<td>09:47</td>
<td>2:00</td>
</tr>
</tbody>
</table>

**stop: 1**
- **chosen TW:**
- **start:** Mon 06:30
- **end:** Mon 07:00

**lateness:** 0:47

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>drive</th>
<th>09:47</th>
<th>12:03</th>
<th>2:16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mon</td>
<td>(un)</td>
<td>12:03</td>
<td>14:03</td>
<td>2:00</td>
</tr>
</tbody>
</table>

**stop: 2**
- **chosen TW:**
- **start:** Mon 10:30
- **end:** Mon 11:00

**lateness:** 1:03

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>drive</th>
<th>14:03</th>
<th>18:17</th>
<th>2:14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mon</td>
<td>rest</td>
<td>16:17</td>
<td>17:32</td>
<td>0:45</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>drive</td>
<td>17:02</td>
<td>17:58</td>
<td>0:56</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>refuel</td>
<td>17:58</td>
<td>18:18</td>
<td>0:20</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>drive</td>
<td>18:18</td>
<td>21:52</td>
<td>3:34</td>
</tr>
<tr>
<td></td>
<td>Mon</td>
<td>rest</td>
<td>21:52</td>
<td>06:52</td>
<td>9:00</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>drive</td>
<td>06:52</td>
<td>11:22</td>
<td>4:30</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>rest</td>
<td>11:22</td>
<td>12:07</td>
<td>0:45</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>drive</td>
<td>12:07</td>
<td>16:37</td>
<td>4:30</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>rest</td>
<td>16:37</td>
<td>01:40</td>
<td>9:03</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>drive</td>
<td>01:40</td>
<td>06:10</td>
<td>4:30</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>rest</td>
<td>06:10</td>
<td>06:55</td>
<td>0:45</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>drive</td>
<td>06:55</td>
<td>08:30</td>
<td>1:35</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>(un)</td>
<td>08:30</td>
<td>10:30</td>
<td>2:00</td>
</tr>
</tbody>
</table>

**stop: 3**
- **chosen TW:**
- **start:** Wed 08:00
- **end:** Wed 08:30

**lateness:** 0:00

<table>
<thead>
<tr>
<th></th>
<th>Wed</th>
<th>drive</th>
<th>10:30</th>
<th>13:05</th>
<th>2:35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wed</td>
<td>refuel</td>
<td>13:05</td>
<td>13:25</td>
<td>0:20</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>drive</td>
<td>13:25</td>
<td>13:45</td>
<td>0:20</td>
</tr>
<tr>
<td></td>
<td>Wed</td>
<td>rest</td>
<td>13:45</td>
<td>01:25</td>
<td>11:40</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>drive</td>
<td>01:25</td>
<td>01:45</td>
<td>0:20</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>rest</td>
<td>01:45</td>
<td>02:00</td>
<td>0:15</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>wait</td>
<td>02:00</td>
<td>03:30</td>
<td>1:30</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>(un)</td>
<td>03:30</td>
<td>05:30</td>
<td>2:00</td>
</tr>
</tbody>
</table>

**stop: 4**
- **chosen TW:**
- **start:** Thu 03:30
- **end:** Thu 04:00

**lateness:** 0:00

<table>
<thead>
<tr>
<th></th>
<th>Thu</th>
<th>drive</th>
<th>05:30</th>
<th>09:40</th>
<th>4:10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Thu</td>
<td>rest</td>
<td>09:40</td>
<td>10:10</td>
<td>0:30</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>drive</td>
<td>10:10</td>
<td>14:40</td>
<td>4:30</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>rest</td>
<td>14:40</td>
<td>15:25</td>
<td>0:45</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>drive</td>
<td>15:25</td>
<td>16:25</td>
<td>1:00</td>
</tr>
<tr>
<td></td>
<td>Thu</td>
<td>rest</td>
<td>16:25</td>
<td>01:25</td>
<td>9:00</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>01:25</td>
<td>05:55</td>
<td>4:30</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>rest</td>
<td>05:55</td>
<td>06:40</td>
<td>0:45</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>06:40</td>
<td>11:10</td>
<td>4:30</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>rest</td>
<td>11:10</td>
<td>11:55</td>
<td>0:45</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>11:55</td>
<td>12:00</td>
<td>0:05</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>refuel</td>
<td>12:00</td>
<td>12:20</td>
<td>0:20</td>
</tr>
<tr>
<td></td>
<td>Fri</td>
<td>drive</td>
<td>12:20</td>
<td>13:12</td>
<td>0:52</td>
</tr>
</tbody>
</table>

**stop: 5**
- **chosen TW:**
- **start:** Mon 00:00
- **end:** Sun 23:59

**lateness:** 0:00

Table 4.6.: Schedule from optimization step 2
### Table 4.7.: Schedule from optimization step 3

<table>
<thead>
<tr>
<th>stop: 1</th>
<th>chosen TW: start:</th>
<th>Mon 06:30</th>
<th>end: Mon 07:00</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lateness:</td>
<td>0.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mon unload</td>
<td>07:47</td>
<td>09:47</td>
</tr>
<tr>
<td></td>
<td>Mon drive</td>
<td>08:47</td>
<td>12:03</td>
</tr>
<tr>
<td></td>
<td>Mon unload</td>
<td>12:03</td>
<td>14:03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stop: 2</th>
<th>chosen TW: start:</th>
<th>Mon 10:30</th>
<th>end: Mon 11:00</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lateness:</td>
<td>1:03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mon drive</td>
<td>14:03</td>
<td>16:17</td>
</tr>
<tr>
<td></td>
<td>Mon rest</td>
<td>16:17</td>
<td>17:02</td>
</tr>
<tr>
<td></td>
<td>Mon drive</td>
<td>17:02</td>
<td>17:58</td>
</tr>
<tr>
<td></td>
<td>Mon refuel</td>
<td>17:58</td>
<td>18:18</td>
</tr>
<tr>
<td></td>
<td>Mon drive</td>
<td>18:18</td>
<td>21:52</td>
</tr>
<tr>
<td></td>
<td>Mon rest</td>
<td>21:52</td>
<td>06:52</td>
</tr>
<tr>
<td></td>
<td>Tue drive</td>
<td>06:52</td>
<td>11:22</td>
</tr>
<tr>
<td></td>
<td>Tue rest</td>
<td>11:22</td>
<td>12:07</td>
</tr>
<tr>
<td></td>
<td>Tue drive</td>
<td>12:07</td>
<td>16:37</td>
</tr>
<tr>
<td></td>
<td>Tue rest</td>
<td>16:37</td>
<td>03:40</td>
</tr>
<tr>
<td></td>
<td>Wed drive</td>
<td>03:40</td>
<td>09:40</td>
</tr>
<tr>
<td></td>
<td>Wed rest</td>
<td>09:40</td>
<td>08:55</td>
</tr>
<tr>
<td></td>
<td>Wed drive</td>
<td>08:55</td>
<td>09:29</td>
</tr>
<tr>
<td></td>
<td>Wed refuse</td>
<td>09:29</td>
<td>09:40</td>
</tr>
<tr>
<td></td>
<td>Wed drive</td>
<td>09:40</td>
<td>10:50</td>
</tr>
<tr>
<td></td>
<td>Wed wait</td>
<td>10:50</td>
<td>11:00</td>
</tr>
<tr>
<td></td>
<td>Wed unload</td>
<td>11:00</td>
<td>13:00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stop: 3</th>
<th>chosen TW: start:</th>
<th>Wed 10:30</th>
<th>end: Wed 11:00</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lateness:</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wed drive</td>
<td>13:00</td>
<td>15:55</td>
</tr>
<tr>
<td></td>
<td>Wed rest</td>
<td>15:55</td>
<td>03:05</td>
</tr>
<tr>
<td></td>
<td>Thu drive</td>
<td>03:05</td>
<td>03:25</td>
</tr>
<tr>
<td></td>
<td>Thu rest</td>
<td>03:25</td>
<td>03:40</td>
</tr>
<tr>
<td></td>
<td>Thu unload</td>
<td>03:40</td>
<td>08:40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stop: 4</th>
<th>chosen TW: start:</th>
<th>Thu 03:30</th>
<th>end: Thu 04:00</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lateness:</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Thu drive</td>
<td>05:40</td>
<td>07:02</td>
</tr>
<tr>
<td></td>
<td>Thu refuse</td>
<td>07:02</td>
<td>07:22</td>
</tr>
<tr>
<td></td>
<td>Thu drive</td>
<td>07:22</td>
<td>10:10</td>
</tr>
<tr>
<td></td>
<td>Thu rest</td>
<td>10:10</td>
<td>10:40</td>
</tr>
<tr>
<td></td>
<td>Thu drive</td>
<td>10:40</td>
<td>15:10</td>
</tr>
<tr>
<td></td>
<td>Thu rest</td>
<td>15:10</td>
<td>15:55</td>
</tr>
<tr>
<td></td>
<td>Thu drive</td>
<td>15:55</td>
<td>16:55</td>
</tr>
<tr>
<td></td>
<td>Thu rest</td>
<td>16:55</td>
<td>01:55</td>
</tr>
<tr>
<td></td>
<td>Fri drive</td>
<td>01:55</td>
<td>06:25</td>
</tr>
<tr>
<td></td>
<td>Fri rest</td>
<td>06:25</td>
<td>07:10</td>
</tr>
<tr>
<td></td>
<td>Fri drive</td>
<td>07:10</td>
<td>11:40</td>
</tr>
<tr>
<td></td>
<td>Fri rest</td>
<td>11:40</td>
<td>12:25</td>
</tr>
<tr>
<td></td>
<td>Fri drive</td>
<td>12:25</td>
<td>12:30</td>
</tr>
<tr>
<td></td>
<td>Fri refuel</td>
<td>12:30</td>
<td>12:50</td>
</tr>
<tr>
<td></td>
<td>Fri drive</td>
<td>12:50</td>
<td>13:42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stop: 5</th>
<th>chosen TW: start:</th>
<th>Mon 00:00</th>
<th>end: Sun 23:59</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lateness:</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.8.: Schedule from optimization step 4

<table>
<thead>
<tr>
<th>day</th>
<th>Step 4</th>
<th>from</th>
<th>until</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>(unload)</td>
<td>07:47</td>
<td>09:47</td>
<td>2:00</td>
</tr>
</tbody>
</table>

#### stop: 1
chosen TW: start: Mon 06:30 end: Mon 07:00
lateness: 0:47

<table>
<thead>
<tr>
<th>Mon</th>
<th>(unload)</th>
<th>09:47</th>
<th>12:03</th>
<th>2:16</th>
</tr>
</thead>
</table>

#### stop: 2
chosen TW: start: Mon 10:30 end: Mon 11:00
lateness: 1:03

<table>
<thead>
<tr>
<th>Mon</th>
<th>drive</th>
<th>14:03</th>
<th>16:17</th>
<th>2:14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>rest</td>
<td>16:17</td>
<td>17:02</td>
<td>0:46</td>
</tr>
<tr>
<td>Mon</td>
<td>drive</td>
<td>17:02</td>
<td>17:58</td>
<td>0:56</td>
</tr>
<tr>
<td>Mon</td>
<td>(unload)</td>
<td>18:18</td>
<td>21:52</td>
<td>3:34</td>
</tr>
<tr>
<td>Mon</td>
<td>drive</td>
<td>18:18</td>
<td>21:52</td>
<td>3:34</td>
</tr>
<tr>
<td>Tue</td>
<td>drive</td>
<td>09:52</td>
<td>11:22</td>
<td>3:30</td>
</tr>
<tr>
<td>Tue</td>
<td>rest</td>
<td>11:22</td>
<td>12:07</td>
<td>0:45</td>
</tr>
<tr>
<td>Tue</td>
<td>(unload)</td>
<td>12:07</td>
<td>13:37</td>
<td>1:30</td>
</tr>
<tr>
<td>Tue</td>
<td>drive</td>
<td>12:07</td>
<td>13:37</td>
<td>1:30</td>
</tr>
<tr>
<td>Tue</td>
<td>rest</td>
<td>13:37</td>
<td>03:37</td>
<td>11:00</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>03:37</td>
<td>09:07</td>
<td>5:30</td>
</tr>
<tr>
<td>Wed</td>
<td>rest</td>
<td>09:07</td>
<td>08:52</td>
<td>0:15</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>08:52</td>
<td>09:26</td>
<td>0:34</td>
</tr>
<tr>
<td>Wed</td>
<td>refuel</td>
<td>09:26</td>
<td>09:46</td>
<td>0:20</td>
</tr>
<tr>
<td>Wed</td>
<td>drive</td>
<td>09:46</td>
<td>10:47</td>
<td>1:01</td>
</tr>
<tr>
<td>Wed</td>
<td>(unload)</td>
<td>10:47</td>
<td>12:47</td>
<td>2:00</td>
</tr>
</tbody>
</table>

#### stop: 3
chosen TW: start: Wed 10:30 end: Wed 11:00
lateness: 0:00

<table>
<thead>
<tr>
<th>Wed</th>
<th>drive</th>
<th>12:47</th>
<th>15:42</th>
<th>2:55</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed</td>
<td>rest</td>
<td>15:42</td>
<td>02:55</td>
<td>11:13</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>02:55</td>
<td>03:15</td>
<td>0:20</td>
</tr>
<tr>
<td>Thu</td>
<td>rest</td>
<td>03:15</td>
<td>03:30</td>
<td>0:15</td>
</tr>
<tr>
<td>Thu</td>
<td>(unload)</td>
<td>03:30</td>
<td>06:30</td>
<td>2:00</td>
</tr>
</tbody>
</table>

#### stop: 4
chosen TW: start: Thu 03:30 end: Thu 04:00
lateness: 0:00

<table>
<thead>
<tr>
<th>Thu</th>
<th>drive</th>
<th>05:30</th>
<th>06:52</th>
<th>1:22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thu</td>
<td>refuel</td>
<td>06:52</td>
<td>07:12</td>
<td>0:20</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>07:12</td>
<td>10:00</td>
<td>2:48</td>
</tr>
<tr>
<td>Thu</td>
<td>rest</td>
<td>10:00</td>
<td>10:30</td>
<td>0:30</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>10:30</td>
<td>15:00</td>
<td>4:30</td>
</tr>
<tr>
<td>Thu</td>
<td>rest</td>
<td>15:00</td>
<td>16:45</td>
<td>1:45</td>
</tr>
<tr>
<td>Thu</td>
<td>drive</td>
<td>16:45</td>
<td>01:45</td>
<td>9:00</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>01:45</td>
<td>06:15</td>
<td>4:30</td>
</tr>
<tr>
<td>Fri</td>
<td>rest</td>
<td>06:15</td>
<td>07:00</td>
<td>0:45</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>07:00</td>
<td>11:30</td>
<td>4:30</td>
</tr>
<tr>
<td>Fri</td>
<td>rest</td>
<td>11:30</td>
<td>12:15</td>
<td>0:45</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>12:15</td>
<td>12:20</td>
<td>0:05</td>
</tr>
<tr>
<td>Fri</td>
<td>refuel</td>
<td>12:20</td>
<td>12:40</td>
<td>0:20</td>
</tr>
<tr>
<td>Fri</td>
<td>drive</td>
<td>12:40</td>
<td>16:30</td>
<td>3:50</td>
</tr>
</tbody>
</table>

#### stop: 5
chosen TW: start: Mon 00:00 end: Sun 23:59
lateness: 0:00
4.8. Numerical experiments - Analysis

In the following, the results of the numerical experiments are presented and analyzed. In Section 4.8.1 the influence of the filter distance on the run time is discussed. Then, in Section 4.8.2, the impact of the number and the length of customer time windows on the run time is considered. Section 4.8.3 gives some managerial insights.

4.8.1. The influence of the filter distance on the run time

Figure 4.6 depicts the overall run times (sum of the run times of optimization steps 1-4) of all test runs\(^\text{76}\) depending on the number of locations. The latter depend, in turn, on the chosen filter distance.\(^\text{77}\) It clearly can be seen that the filter distance and thus the number of locations considered very strongly influences the run times. The figure shows three test runs with an overall run time of more than 30 minutes and 20 test runs (1.5%) with a run time between 25 and 30 minutes. The other test runs (98.3%) required less than 25 minutes.

![Run times depending on the number of locations including gas stations and filter distances](image)

---

\(^{76}\) For each test instance, 6 test runs were performed that differ by the chosen filter distance (see Section 4.6).

\(^{77}\) The detailed results can be found in Appendix B.
The three test runs with a run time of more than 30 minutes belong to the cases with a filter distance of 100 km and have the largest number of locations to be considered. In the case of the longest run time of about 58 minutes (3509 sec) (base instance 14, 3 time windows, each with a length of 120 minutes), it took nearly 58 minutes (3464 sec) in the second step until a first feasible solution was found. In most of the cases in which in at least one of the steps no optimal solution was found the second step was the bottleneck. But in some of those cases with many locations the first optimization step was already very expensive. In the two other cases where the overall run time exceeded 30 minutes, the first step was the most time consuming. In one of the two cases (base instance 8, 3 time windows with a length of 600 minutes) the first step took 32 minutes. In the other case (base instance 15, 3 time windows, each with a length of 600 minutes), more than 21 minutes were needed to obtain an optimal solution in this step (overall run time: 36 minutes). In the test run with a run time of exactly 30 minutes (base instance 8, two time windows, each with a length of 600 minutes), only feasible solutions were found in each step with a run time for step one of 25 minutes. For the test run (base instance 14, 3 time windows, each with a length of 600 seconds) with a run time very close to 30 minutes (1780 sec), an optimal solution was found in each step. For the other test runs with an overall run time of more than 25 minutes and a filter distance of 100 km, optimization step 2 had the longest run time by far. This occurred for 6 instances from base instance 15 and one instance from base instance 14. For each of these instances, steps 1 and 2 required in total more than 25 minutes.

For a filter distance of 200 km, in step two the time limit was reached 4 times, for a filter distance of 300 km it was reached 3 times and for the filter distances of 400 and 500 km it was reached 2 times in each case. All of the corresponding instances were derived from base instance 15. Considering the filter distance of 1000 km, overall run times were below 3 minutes with an average run time of 11 seconds.

In total, 27.38 hours were needed for all 1350 test runs. Figure 4.7 shows the proportions of the different optimization steps on the overall run time for all test instances. The greatest impact has optimization step 2 with a cumulative duration of 18.44 hours (67%). Optimization step 1 took 5.76 hours (21%) to be completed. Time limits for step 3 were only relevant in two cases (the second and third case with a run time of more than 30 minutes described above). The overall duration of 1.82 hours (7%) is rather short which shows that the refueling subproblem is much faster to solve than the subproblem for planning time windows, driving times, rest periods and breaks. The last optimization step took 1.36 h in total (5%).

Figures 4.8 and 4.9 show the proportions of the different steps on the overall (for all test instances) run time for a filter distance of 100 km and a filter distance of 1000 km, respectively.

When reducing the filter distance, the proportions for the different optimization steps are comparable in their magnitude although the run times increase significantly. Note that the overall run time for all instances was 14.05 h if a filter distance of 100 km was chosen and only 0.71 h if a filter distance of 1000 km was selected. This means that there is a 95% decrease of the overall run time with a filter distance of 1000 km compared to 100 km.
Figure 4.7.: The proportions of the different optimization steps on the overall run time

Figure 4.8.: The proportions of the different optimization steps on the overall run time for a filter distance of 100 km (225 test instances)

Figure 4.9.: The proportions of the different optimization steps on the overall run time for a filter distance of 1000 km (225 test instances)
Figures 4.10 and 4.11 depict the average run times per base instance and per optimization step for the filter distances of 100 km and 1000 km.

Figure 4.10.: Average run times per base instance and per optimization step for a filter distance of 100 km

Figure 4.11.: Average run times per base instance and per optimization step for a filter distance of 1000 km

Again, it can be seen that on average the run time of step two is the longest with the two exceptions for base instances 2 and 8 and filter distance 100 km. Note that the large values for optimization step 1 and a filter distance of 100 km of base instance 8 are mainly due to the two cases described before (see pages 199 et seq.). In total, there are three base
instances for which the run time of step 1 is longer than that of step 2. For base instance 2, 11 out of 15 test instances (73%) show a longer run time for step 1. In total, there were 263 test runs (19%) for which step 1 took longer than step 2. In those cases, the shorter run time of step 2 may be attributed to the quality of the upper bound provided by step 1.

Although the run times of step 3 only account for 7% of the overall run time for all test runs, the strong growth of the run time with the number of locations, especially when considering the maximum run time for a given number of locations, can be observed here as well (see Figure 4.12).

![Figure 4.12.: Run times of step 3](image)

### 4.8.2. The influence of the number and length of time windows on the run time

It is interesting to see that each of the three instances with a run time of more than 30 minutes has three alternative time windows for each customer stop (see Figure 4.13). This suggests that the number of time windows influences the run time.

A detailed analysis of the average run times depending on filter distances and the number of time windows (see Figure 4.14) did not provide a clear picture. While the longest run times for the filter distances of 100, 400, 500 and 1000 km were obtained for instances with three alternative time windows per customer stop, for the filter distances of 200 and 300 km the instances with three time windows show on average the shortest run times.
4.8. Numerical experiments - Analysis

Figure 4.13.: Run times depending on the number of locations and the number of time windows

Figure 4.14.: Average run time depending on the filter distance and the number of time windows
We observe that the longest run time was obtained for an instance with a time window length of 120 minutes (see Figure 4.15), followed by five instances with a time window of 600 min.

![Graph showing run times depending on the time window (TW) length and on the number of locations.](image)

Figure 4.15.: Run times depending on the time window (TW) length and on the number of locations

Figure 4.16 shows the average run times per filter distance and per time window length. It can be seen that for each filter distance the average run time for instances with time windows of 600 min is always ranked first or second when considering the longest run times. For a time window length of 120 min, significant longer average run times than for the time window lengths of 0, 30 and 60 min can be found for the filter distances 100, 400 and 500 km.
4.8. Numerical experiments - Analysis

Figure 4.16.: Average run times depending on the filter distance and the time window length

4.8.3. Managerial insights

In this section, valuable managerial insights are drawn from the results obtained. First, the solution quality is analyzed depending on the selected filter distance. Afterwards, general findings are discussed that may be relevant for practitioners.

Filter distance and solution quality

Figure 4.17 shows that the average total lateness is constant for the filter distances of 200, 300, 400 and 500 km. For a filter distance of 1000 km there is an increase of 0.08%, i.e. the difference in the overall lateness for all 225 test instances is 35 minutes, that means on average 9 seconds per instance. The increase in lateness stems from two instances. The first one has a total lateness of 18 minutes and is derived from base instance 11. Moreover, it has one time window per customer location with a time window length equal to zero. The other one also refers to base instance 11 with one time window per customer location but with a time window length of 30 minutes. For these two cases, even without considering the time for refueling, time windows cannot be met. The increase in lateness when considering a filter distance of 100 km can be explained by the non-optimal solution values obtained for the instances where time limits were reached and the solution process was stopped prematurely (see Figure 4.4 for the solution process and corresponding time limits, and Section 4.8.1 for a description of test instances with non-optimal solution values).
Fuel costs increase marginally with the filter distance. The difference between choosing a filter distance of 100 km and a filter distance of 1000 km is on average 38 ct (0.04%). Over all 225 instances, this amounts to 86 €. This means that the selection of a large filter distance has the advantage of reducing the problem size and thus the run time without negatively affecting the total fuel cost.

![Graph showing lateness and refueling cost depending on the filter distance](image)

Figure 4.17.: Lateness and refueling cost depending on the filter distance

The average completion time is relatively constant and lies between 107.76 h (with a filter distance of 300 km) and 107.84 h (with a filter distance of 100 km) per week (see Figure 4.18).

Table 4.9 summarizes the findings considering four criteria for each of the filter distances. The filter distance with the best obtained value cumulated over all instances is chosen as reference. The corresponding value is set to be equivalent to 100 %.

It can be seen that slight increases in the solution quality correlate with a significant growth of the run time for the shorter filter distances. When considering the filter distance 100 km, the instances have the largest sizes and thus the run times have the longest durations. Compared to the filter distance 200 km there are more instances for which no optimal solution is found in one or more of the optimization steps. Thus, a deterioration of lateness, fuel cost and completion time can be observed.
Figure 4.18.: Average completion time and filter distance

<table>
<thead>
<tr>
<th>filter distance (km)</th>
<th>% of minimum lateness</th>
<th>% of minimum fuel costs</th>
<th>% of minimum completion time</th>
<th>% of minimum run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100.11%</td>
<td>100.01%</td>
<td>100.07%</td>
<td>2057.62%</td>
</tr>
<tr>
<td>200</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>580.88%</td>
</tr>
<tr>
<td>300</td>
<td>100.00%</td>
<td>100.03%</td>
<td>100.00%</td>
<td>454.55%</td>
</tr>
<tr>
<td>400</td>
<td>100.00%</td>
<td>100.03%</td>
<td>100.01%</td>
<td>379.70%</td>
</tr>
<tr>
<td>500</td>
<td>100.00%</td>
<td>100.04%</td>
<td>100.01%</td>
<td>288.29%</td>
</tr>
<tr>
<td>1000</td>
<td>100.08%</td>
<td>100.04%</td>
<td>100.03%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Table 4.9.: Solution quality and run times depending on the filter distance
4. The combined problem

General findings

In this section, more general findings are discussed. The first table (Table 4.10) depicts the average detour distance\(^{78}\) depending on the chosen filter distance. It can be noted that the best value is obtained for a filter distance of 1000 km. A reason for this may be the consideration of country prices. As prices often do not vary over long distances traveled, the detour distance in many cases is the criterion that eliminates gas stations within the given filter distance. With a filter distance of 1000 km more gas stations are eliminated than with a filter distance of 200 km, thereby removing more gas stations with large detour distances. As the choice of gas stations with large detour distances is reduced, it is quite probable that the optimal solutions for a filter distance of 1000 km incorporate shorter detour distances than those for a smaller filter distance. But this does not always have to be the case.

<table>
<thead>
<tr>
<th>filter distance (km)</th>
<th>(\bar{D}) detour distance (km)</th>
<th>% of minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6.34</td>
<td>119%</td>
</tr>
<tr>
<td>200</td>
<td>5.64</td>
<td>106%</td>
</tr>
<tr>
<td>300</td>
<td>5.37</td>
<td>101%</td>
</tr>
<tr>
<td>400</td>
<td>5.45</td>
<td>102%</td>
</tr>
<tr>
<td>500</td>
<td>5.37</td>
<td>101%</td>
</tr>
<tr>
<td>1000</td>
<td>5.34</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 4.10.: Filter distance and average detour distance

Tables 4.11, 4.12 and 4.13 show the number of refueling stops depending on the filter distance, the number of time windows per customer location and their length.\(^{79}\)

It can be seen that for filter distances of 200 km and 300 km the average number of refueling stops is 2.12. For filter distances of 400, 500 and 1000 km the average number is about 2.19, which is 3% more than the minimum value. For a filter distance of 100 km, the effects of non-optimal solutions have to be taken into account (recall Table 4.9). If time windows are not chosen optimally and thus more lateness is accepted, this leaves more time for additional refueling stops. This may explain why the average number of refueling stops is higher for a filter distance of 100 km than for a filter distance of 200 km.

The number of refueling stops lies between 1 and 4 for all test runs. Considering an arbitrary instance, the difference of the number of refueling stops between two filter distances is at most 1. On average and over all test runs 2.16 refueling stops are made. With more time windows per customer location there are more possibilities for planning arrival times without worsening lateness and the schedule duration. This leaves more freedom and time for refueling and may be the reason why in Table 4.12 the average number of refueling stops slightly increases with the number of time windows per customer location.

\(^{78}\) The average is taken over all instances where per instance the whole detour for all visited gas stations is considered.

\(^{79}\) Note that the number of refueling stops was only considered in the objective function of optimization step 4.
<table>
<thead>
<tr>
<th>filter distance (km)</th>
<th>Ø refueling stops</th>
<th>% of minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2.18</td>
<td>103%</td>
</tr>
<tr>
<td>200</td>
<td>2.12</td>
<td>100%</td>
</tr>
<tr>
<td>300</td>
<td>2.12</td>
<td>100%</td>
</tr>
<tr>
<td>400</td>
<td>2.19</td>
<td>103%</td>
</tr>
<tr>
<td>500</td>
<td>2.19</td>
<td>103%</td>
</tr>
<tr>
<td>1000</td>
<td>2.19</td>
<td>103%</td>
</tr>
</tbody>
</table>

Table 4.11.: Filter distance and the number of refueling stops

<table>
<thead>
<tr>
<th># TW</th>
<th>Ø refueling stops</th>
<th>% of minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.14</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>2.17</td>
<td>101%</td>
</tr>
<tr>
<td>3</td>
<td>2.19</td>
<td>102%</td>
</tr>
</tbody>
</table>

Table 4.12.: The number of time windows and the number of refueling stops

For a time window length of zero, an additional refueling stop will more easily lead to lateness than for a time window length greater than zero. Therefore, the average number of stops for those instances is the lowest (see Table 4.13). For time window lengths of 30, 60 and 120 minutes, the average number of refueling stops is relatively constant. When considering a time window length of 600 minutes, which in reality corresponds to opening hours for many of the considered instances, this is nearly the same as planning arrival times freely without time windows. This means that additional refueling stops have a relatively direct impact on the overall completion time, which is a component of the objective functions of optimization steps 1, 2 and 4. This may be the reason why the average number of refueling stops for a time window length of 600 minutes is less than for time window lengths of 30, 60 and 120 minutes.

<table>
<thead>
<tr>
<th>length TW (min)</th>
<th>Ø refueling stops</th>
<th>% of minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.13</td>
<td>100.0%</td>
</tr>
<tr>
<td>30</td>
<td>2.18</td>
<td>102.1%</td>
</tr>
<tr>
<td>60</td>
<td>2.18</td>
<td>102.1%</td>
</tr>
<tr>
<td>120</td>
<td>2.17</td>
<td>101.9%</td>
</tr>
<tr>
<td>600</td>
<td>2.16</td>
<td>101.2%</td>
</tr>
</tbody>
</table>

Table 4.13.: Time window length and the number of refueling stops
4.9. Possible Heuristic Approaches

This thesis primarily focuses on developing mathematical formulations for challenging real-world transportation problems. Nevertheless, in Section 3.7.1, it is shown how to integrate the MILP for the sequence vehicle refueling problem with time windows into a heuristic for the VRPTW. Moreover, in Section 2.8 a myopic algorithm is presented for the planning of drivers’ driving times, rest periods and breaks and the choice of customer time windows.

In the previous section, we have seen that with a good parameter choice for the filtering algorithm presented in Section 4.5 that eliminates unattractive gas stations, the computational effort of the combined problem can be reduced significantly. As a result, the CPLEX optimization solver delivered good solutions within a reasonable time for the test instances provided. But some companies may be reluctant to deploy a commercial solver due to cost reasons. Additionally, the problem complexity and thus the computational time increase with the number of customer locations and gas stations considered. For problem settings with more than 10 customers planned to be visited during the week, the run duration may not be acceptable in practice. Thus, in this section some ideas are presented to set up a heuristic for the combined problem.

Metaheuristics proposed in the literature provide several frameworks that may be applied for the development of a concrete heuristic for a problem. They guide the search process and often find good solutions with less computational effort than optimization algorithms or simple heuristics (Blum and Roli (2003)). In the following, the main elements are proposed that together with metaheuristic strategies can be used to develop a heuristic.

The basic idea is that the choice of gas stations for refueling is guided by a heuristic scheme. Given that refueling is assumed to take constant time like it is done in Sections 3 and 4, the remaining problem decomposes into two parts:

- the scheduling of driving times, breaks and rest periods, and the choice of customer time windows
- solving the sequence vehicle refueling problem without time aspects

The charm of choosing the gas stations externally by the heuristic is that there are no interdependencies between the two remaining subproblems anymore. Detour consumptions and driving durations to and from gas stations are given by the gas station selection made.

The first subproblem from above can then be solved by a variant of the myopic algorithm of Section 2.8. Therefore, the gas station locations are added as vertices to the problem definition. The algorithm then iterates over the extended list of locations. For the path lengths between each pair of consecutive vertices, detour durations to and from gas stations are added accordingly for each vertex that corresponds to a gas station. This can already be included in the input data. Refueling is assumed to take place in each of the additional vertices. The graph structure then reduces to that shown in Figure 4.19.
Customer locations, start and end locations are shown as large vertices, whereas gas stations are illustrated as small vertices.\textsuperscript{80}

![Graph structure for the extended myopic algorithm](image)

Figure 4.19.: Graph structure for the extended myopic algorithm

When iterating over the list of consecutive locations (and paths between them), if the next vertex represents a customer location, the steps "Scheduling activities on arc \((i, i+1)\)", "Choose time window at stop \(i+1\)" and "Modify rest durations and schedule activities at stop \(i+1\)" are executed one after the other like in the original version of the myopic algorithm (see Figure 2.21 on page 116). In case the next vertex is associated with a gas station, the last two steps are omitted and only the step "Scheduling activities on arc \((i, i+1)\)" is executed. When arriving at a gas station, it is only tested whether the time left until the next daily rest period suffices to refuel. A daily rest period is taken if not enough time is left. If the next daily rest period is planned to be a reduced one, this is taken into account as well as a partial daily rest period taken since the last full daily rest period. Then, the refueling activity is scheduled with constant time.

The methods

- `getNonReducableRestPeriods()
- `reduceRestDurationLastRest()` and
- `extendRestDurationLastRest(< extension in min. >)`

refer to two lists, the list with all daily rest period positions and the list with regular daily rest period positions (positions of daily rest periods that may still be reduced)\textsuperscript{81}. In the original version, the lists are reset after each iteration of the algorithm. For the extended version which includes gas stations, the lists are only set back, if the following arc starts at a customer location.

In a first step, one may consider driving time extensions only if the next location to be visited is a customer location.\textsuperscript{82} This may not have a high influence on the solution, especially if vehicles with large tank capacities are considered and customer stops are more frequent than refueling stops.

\textsuperscript{80}The algorithm chosen for the refueling subproblem may determine a refueling quantity of 0 for a subset of the gas stations selected. In this case, one may choose to remove the gas stations with zero refueling and restart the solution process for this subproblem.

\textsuperscript{81}For more details, see Section 2.8.

\textsuperscript{82}In the original algorithm for the decision whether to extend driving time or not, the first reachable time windows with and without taking a driving time extension are determined for the next customer location which is represented by the next vertex in the list. If the next vertex is a gas station, this is not possible in the same way. If in that case the reachability of time windows should be considered for the next customer location this would mean to look ahead several vertices.
In the heuristic in which the algorithms described in this section may be embedded, gas stations are explicitly chosen for refueling. Thus, consumptions for detours are directly included between each pair of consecutive locations. The consumption between locations $i$ and $i+1$ is given by $\bar{\Delta}^{cons}_{(i,i+1)}$. As mentioned before, with the gas stations chosen for refueling being fixed and assuming that the refueling duration is constant, i.e. does not depend on the amount refueled, there are no time aspects to consider anymore for the part of the problem that deals with refueling decisions. Thus, customer locations can be neglected and the locations considered only consist of all gas station locations chosen by the heuristic, along with the origin and destination locations. Through hiding the customer locations to be visited along the route, the graph structure reduces to the one illustrated in Figure 4.20.

![Figure 4.20: Graph structure for the refueling subproblem (gas stations are represented by 1,2,3,...)](image)

The problem itself is similar to the fixed route vehicle refueling problem considered by Lin et al. (2007) and the fixed-path gas station problem described by Khuller et al. (2007). Diverging from our considerations, in their problem definitions all locations are gas stations whereas in our subproblem no refueling may take place at the start vertex and the end vertex. Additionally, in Chapters 3 and 4 we considered a minimum fuel level to be maintained as a reserve amount at all times, a minimum end fuel level, and a minimum purchase quantity. These aspects are neglected in the algorithms proposed by Lin et al. (2007) and Khuller et al. (2007) who also do not describe how to deal with an initial fuel level different from zero. How their algorithms can be modified to include these aspects is described in the following.

In Chapters 3 and 4 we require a minimum amount of fuel to be maintained in the tank at all times $\bar{T}^{min}$ to not run out of gas in case of unforeseen events, diverging fuel consumption rates, or due to inaccuracies in the fuel level indicator. To take care of this reserve amount, we subtract it from the tank capacity $\bar{T}^{max}$ ($\bar{T}^{max} = \bar{T}^{max} - \bar{T}^{min}$), the initial fuel level $\bar{f}^{start}$ ($\bar{f}^{start} = \bar{f}^{start} - \bar{T}^{min}$) and the end fuel level $\bar{f}^{end}$ ($\bar{f}^{end} = \bar{f}^{end} - \bar{T}^{min}$) of the original problem, and solve the problem with the modified parameters.

While the amount of fuel left when arriving at the final destination is of no concern in Lin et al. (2007) and Khuller et al. (2007), we want to be able to consider a minimum end fuel level. This can be integrated by adding the modified end fuel level $\bar{f}^{end}$ to the fuel consumption for the path between the last gas station and the final destination. Then, the end fuel level can again be neglected in the modified problem definition, that means the minimum end fuel level is indirectly set to be zero. The solution obtained is equivalent

---

[83] As refueling costs are minimized, this means that the vehicle either reaches the final destination with an empty tank or, in case no refueling is necessary, with the initial fuel level minus the fuel consumption for the complete route.
4.9. Possible Heuristic Approaches

to the solution to the original problem. The end fuel level can be determined by adding
the original end fuel level to the end fuel level of the solution obtained for the modified
problem.

In both algorithms, the one of Lin et al. (2007) and the one of Khuller et al. (2007),
the refueling costs for the start location can be set to infinity (or a very large value) to
avoid refueling there. If the start vertex is still chosen for refueling during the algorithm
execution, no solution exists to the original subproblem, as the first gas station along the
route is not reachable.

In Khuller et al. (2007) it is assumed, without loss of generality, that the vehicle starts
with a zero fuel level. In case that refueling is necessary to reach the final destination\(^{84}\), a
start fuel level of \(f_{\text{start}}\) can be included in this algorithm, still maintaining the possibility
to decompose the refueling problem into independent subproblems considering the "break-
points" of the problem\(^{85}\). It is required to add an artificial gas station at the beginning
of the location list with zero fuel costs and a fuel consumption of \(T_{\text{max}} - f_{\text{start}}\) for reaching the
original start location. The refueling determined at the artificial gas station is neglected
to obtain the solution to the original problem.

A possibility for introducing a minimum purchase quantity will be described in the following
for the algorithm presented by Lin et al. (2007). The linear-time greedy algorithm of
Lin et al. (2007) computes for every gas station \(i\) which gas station following \(i\) is the
farthest reachable gas station \(\text{FAR}(i)\) with a fuel level equal to the maximum tank capacity.
Additionally, it determines for each gas station \(i\) the next gas station \(\text{LOW}(i)\) in the
sequence with a lower fuel price than \(i\). In the original version, with the information above
determined for each gas station (except for the final one) a refueling plan is set up iterating
over the sequence of gas stations. Upon reaching a gas station, there are two possibilities
to consider: either the next gas station in the sequence with cheaper fuel price is reachable
with a full tank (\(\text{LOW}(i) \leq \text{FAR}(i)\)) or it is not (\(\text{LOW}(i) > \text{FAR}(i)\)). In the latter
case the tank is refilled completely and the vehicle moves to the next gas station in the
sequence. The fuel consumption to reach this gas station is subtracted from the fuel level.
In the other case, the fuel consumption to the next cheaper gas station is determined.
Here, again two cases are possible: either refueling is necessary to reach the subsequent
gas station with lower fuel price or the fuel in the tank suffices to reach this gas station. If
refueling is necessary, the refueling amount is chosen as small as possible. When reaching
the the next gas station with cheaper fuel price the tank should be empty. In either of the
two cases, the vehicle moves to the next cheaper gas station \(\text{LOW}(i)\) and the fuel level
\(\text{fuelInTank}\) is updated accordingly.

Figure 4.21 shows the proposed modified version of the algorithm of Lin et al. (2007)
respecting the minimum purchase quantity in a heuristic manner. The modifications are
marked with green color. Similar as in the original version, we determine \(\text{LOW}(i)\) and
\(\text{FAR}(i)\) for each gas station \(i \in \{0, 1, \ldots, n - 2\}\). The algorithms to determine \(\text{LOW}(i)\)
and \(\text{FAR}(i), i \in 0, 1, \ldots, n - 2\) can be found in the appendices (Appendix C) as well as
the pseudo code of the other algorithms described in the following.

\(^{84}\) It can easily be tested beforehand if refueling is necessary.

\(^{85}\) For more details see Khuller et al. (2007).
Figure 4.21.: Algorithm for the fixed route refueling problem respecting the minimum purchase quantity
Diverging from Lin et al. (2007), we compute for each $i \in \tilde{S}_{\text{stations}}$ the fuel consumption to reach the next gas station with lower fuel price $\text{consLOW}(i)$ in a separate algorithm (Algorithm 14 in Appendix C) with complexity $O(n)$, where $n$ is the number of gas stations.

Instead of refueling the amount determined by the original version (this amount is denoted by $\text{helpRefuel}$ in Figure 4.21), it is tested in advance if the amount is greater than or equal to the minimum purchase quantity $\Delta_{\min}$. In that case, the algorithm proceeds as in the original version. Otherwise, a modification step is performed. This modification step considers the current gas station and potentially the last gas station with positive refueling amount. Successively, it is tried to

- refuel the minimum purchase quantity and go to the same gas station as would be chosen by the original version. This may not be possible due to the tank capacity.
- leave out refueling at this gas station $i$ and go to the next gas station $i + 1$ (even if $\text{LOW}(i) \leq \text{FAR}(i)$). This is not possible if the current fuel level does not suffice to reach $i + 1$ without refueling.
- modify refueling at the preceding gas station with positive refueling amount
  - to be able to exactly refuel the minimum purchase quantity at the current gas station. This is not done if reducing the refueling amount at the preceding gas station accordingly would lead to a refueling amount less than the minimum purchase quantity.
  - by setting the refueling amount to zero. This amount is instead refueled additionally to $\text{helpRefuel}$ at the current gas station.

The choice of the gas station to be considered in the following iteration is made in an analogous manner to the original version. It may happen that no refueling has been planned at any gas station prior to the current one. In that case no solution to the problem can be found respecting the minimum purchase quantity. The algorithm is stopped as no feasible solution exists if the minimum purchase quantity has to be respected.

To reach the final destination with a fuel level equal to the minimum end fuel level, an additional modification is proposed. Details on this and the requirements on the tank capacity as well as the complete algorithm can be found in Appendix C.

As mentioned before, the idea is that the heuristic in which the algorithms described in this section are embedded controls the choice of gas stations. To efficiently guide the search process, metaheuristics can serve as templates. It is important that a metaheuristic applied to an optimization problem needs to find a balance between diversification (exploration) and intensification (exploitation) to be successful. Metaheuristics primarily differ in the way how they try to reach this goal (Birattari et al. (2001)). While basic single-solution-based metaheuristics emphasize exploitation, basic population-based metaheuristics concentrate more on exploration (Boussaïd et al. (2013)). Single-solution-based metaheuristics manipulate a single solution at each stage, whereas population-based metaheuristics manipulate a collection of solutions. If a single-solution based metaheuristic like for example Simulated Annealing (Kirkpatrick et al. (1983)) or the Threshold Accepting Method (Dueck and Scheuer (1990)) is chosen, the initial solution can either be generated by randomly selecting a set of gas stations or by using a construction heuristic.

---

86 This is never possible in case $\text{LOW}(i) > \text{FAR}(i)$. 

A simple approach for a construction heuristic is to always refill completely and only stop at a gas station if no gas station with a closer distance to the final destination is reachable with the remaining fuel amount in the tank. A possible implementation that integrates the consideration of detour consumptions as well as the aspects described above (including the minimum purchase amount) is given with Algorithm 16 in Appendix C. Gas station prices are ignored by the heuristic. The underlying graph structure resembles the one described by Suzuki (2008, 2009) (see the upper part of Figure 3.3 on page 152), ignoring customer locations. With the selection of gas stations obtained, the construction heuristic has to be completed by solving the subproblem of scheduling driver rest periods and breaks and choosing time windows. For this purpose, the modified myopic algorithm described at the beginning of this section can be applied.

For local search steps, neighborhood relations have to be defined accordingly. The neighborhood exploration then starts from a candidate solution and iteratively moves to a neighboring solution that may differ for example by one or more gas stations. Gas stations to add or to remove from a solution can be chosen completely randomly or by systematically choosing gas stations, for example, to be exchanged between a particular pair of consecutive customers. This may, for instance, be advantageous to intensify the search if in the last step lateness is reduced by adding a gas station between such a pair of customers. Different add, remove or exchange steps of one or multiple gas stations may be combined. A lower bound on the overall number of gas stations to be considered in each step can be given by dividing the overall route consumption (without detours) plus the difference between the end fuel level and the start fuel level by the (modified) tank capacity and rounding up to the nearest integer \( \left\lceil \frac{\sum_{k=0}^{n-1} \Delta c_{k+1} + f_{\text{end}} - f_{\text{start}}}{T_{\text{max}}} \right\rceil \). Restricting the number of refueling stops may help raise driver acceptance. Therefore, a corresponding upper bound may be defined, for example, by multiplying the lower bound with a factor greater than one.

Note that the combined problem is a multicriteria optimization problem. This means that different values have to be compared to determine the solution quality and to decide whether a solution obtained in a step is taken into account or if it is discarded by the heuristic. As already mentioned in the previous sections, we consider punctuality to be the most important criterion. Therefore, minimizing lateness has highest priority. Considering overall schedule duration and refueling costs, the tradeoff has to be taken into account. Depending on user preferences, the number of gas stations in the solution may be considered as a fourth criterion.
5. Summary and future research

In this last section, we reflect on the main results of this study and reveal aspects that are worthwhile to be discussed and analyzed in future research.

5.1. Summary

Transport companies face growing pressure because of converging cost structures in the EU and increasing just-in-time management practices. Fuel is a main cost driver in the European road haulage sector. EU legislation on driving times, rest periods and breaks has a high influence on the arrival times at customer locations and on the travel durations in general. Thus, the need to keep transport costs low and to satisfy customer demands on time requires the consideration of fuel expenditures and driving times, rest periods and breaks when determining a driver schedule.

Regulation (EC) No 561/2006 defines the rules for the number, duration and time intervals when rest periods and breaks have to be taken with the objective to ensure road safety, adequate working conditions, and undistorted competition in the road haulage sector. A strict adherence to the rules is necessary, since depending on the seriousness, infringements may have severe consequences for the drivers and the transport company itself. Strongly varying diesel prices across different European countries allow a high cost saving potential, especially when considering long-haul trips that involve crossing several national borders. We described why it is not recommendable to consider fuel optimization as an isolated problem. When choosing appropriate customer time windows and determining driver rest periods and breaks, the time needed for refueling stops has to be taken into account as otherwise the time needed for an unplanned refueling may lead to changes in the driver schedule and thus jeopardize a punctual arrival. When planning refueling stops, it is required, among other things, to determine between which pairs of consecutive customer locations refueling should be done not only considering fuel costs as decision criterion but also minimizing lateness.

A short process analysis conducted on-site at a medium-sized company operating transport services in Europe gave detailed insights into the planning processes in practice and revealed possibilities for decision support for dispatchers and drivers. The different components and interfaces to acquire up-to-date data and the incorporation of the developed models and algorithms into a transport management software were described and a service oriented architecture was proposed.
Before considering the combined problem, the two subproblems, the first one addressing driver rest periods and breaks and the second one considering optimal refueling, are at first studied as isolated problems.

At first, a MILP model and optimization strategies were proposed that, together with a transformation algorithm (see Chapter 2.6), allow to plan driver activities in compliance with Regulation (EC) No 561/2006 for a given sequence of customer locations and other stops to be visited by a single-manned vehicle. Each customer location has one or multiple time windows among which a choice has to be made. The incorporation of daily rest periods and breaks allows for greater planning reliability. A special feature is the consideration of "soft" time windows which has not been studied in this context so far. By penalizing lateness in the objective function instead of prohibiting the arrival after the end of the time window, schedules are found even if lateness cannot be avoided. The resulting schedule gives important information to the dispatcher that is necessary to set up a better schedule. In online re-planning, lateness can be revealed at an early stage such that it is possible to reorganize the schedule or to negotiate arrival times with customers before communication effort and costs increase, and further delays or cancellations are unavoidable. In this way, transport undertakings as well as their customers can benefit and pressure on dispatchers and drivers can be reduced. Attempts to choose time windows that are not reachable can be avoided.

Test instances were derived from real data provided by a German haulage company that operates vehicles in Europe. Vehicle routes were reconstructed for one week, involving between 2 and 10 customer locations and stops for start and end locations (i.e. in total, 4 to 12 stops). Arrival times planned by the dispatchers were used as a basis to generate different time windows. The number of time windows and their length were varied to obtain different test instances. We examined the run time, lateness and completion time for all of the instances depending on the number of stops, and the number and length of time windows. For all of the instances, reasonable run times were achieved ranging from 0.03 seconds (4 stops, 3 time windows, time window length: 10 h) to 10.94 seconds (11 stops, 3 time windows, time window length: 10 h) for the MILP model with consideration of the optional rules on a desktop computer, where the number of stops had the most influence on the run time.

The optional rules were deactivated in the original MILP model to test their influence on the above criteria. The run time was reduced significantly by between 37 % (base instance 1, 4 stops) up to 81 % (base instance 15, 12 stops). On the other hand, the overall lateness was 55 % less if the optional rules were allowed and the schedule duration was reduced by 5 %.

The proposed MILP models allow to establish an optimal driver schedule with the help of optimization software. As the use of a commercial solver can be an obstacle for a company due to cost reasons, we wished to investigate the magnitude of the schedule improvement compared to a heuristic that should simulate the usage of sophisticated strategies by an experienced dispatcher. Based on the idea of a driver status that is modified with each new activity, a myopic algorithm was developed that can only "see" the route until the next customer stop and the corresponding customer time window in advance, and plans driver activities accordingly. Simple strategies were chosen to also integrate the optional rules.
The myopic algorithm achieved 18% overall lateness reduction and no increase in the average schedule duration in comparison with the model without optional rules. Together with the run time of the myopic algorithm that was less than 1 millisecond, the algorithm itself is interesting. The main advantages of the myopic algorithm are its short run time and that no optimization solver is necessary to obtain a solution. The short-sightedness and concentration on one arc at a time makes resulting schedules easy to understand.

Similar to the models, the possibility to start with a given driver status allows for online re-planning. The consideration of the complete tour with all stops allows to construct a schedule with globally optimized lateness (the most important criterion) and completion time when solving the MILP model with optional rules. The overall lateness was 45% less compared to the myopic algorithm and the average schedule duration was reduced by 3.5%. The run time is longer, but depending on the fleet size, the length of the planning horizon and the available computing capacity, online re-planning still may be considered.

The largest advantage of the MILP models is the simultaneous consideration of all stops. If the customer locations to be visited in the considered week are not known in advance but only for the next one or two stops, the dispatcher has to choose among different opportunities without exactly knowing future requests. This reduces the benefit of the MILP model with optional rules.

When regarding the refueling subproblem, again a fixed sequence of customer locations with time windows was considered for a single vehicle. Among the different graph structures analyzed we chose the one that considers detour distances to gas stations from the main route and back. We mentioned reasons to keep to the main route and not to include routing decisions between customer locations on the basis of gas station prices and detours to gas stations. On the basis of the standard fuel optimizer model described by Suzuki (2008, 2009), a MILP model was developed integrating (hard) time windows and multiple customer locations to be visited. In a short digression we showed how to integrate vehicle refueling into the VRPTW extending the Solomon II heuristic. In order to have a test bed, we enriched the well-known Solomon benchmark instances with locations of gas stations, their prices and other necessary parameters for the refueling model. In total, 336 instances were solved and the results obtained indicate that the tour length moderately increases as the spread in fuel prices becomes larger. We also reported on run time and showed that the maximal computational effort is a function of the mean tour length in the solution.

On the basis of the MILP model considering driving times, rest periods, breaks and vehicle refueling we developed a MILP model that plans driver activities in accordance with Regulation (EC) No 561/2006 simultaneously considering the choice of customer time windows, refueling stops and refueling quantities. For the combined problem, we again considered "soft" time windows to find solutions if lateness cannot be avoided completely. The main objectives were the minimization of overall lateness, completion time and fuel expenditures. The solution process presented to solve the resulting multicriteria optimization problem consists of four optimization steps and a transformation algorithm that is needed to obtain a readable driver schedule.

We observed that often list prices were constant over several days. As future prices currently cannot be predicted exactly several days in advance, the approach to plan with the
current price is justifiable. However, online re-planning is recommended and this step can be carried out with the solution process presented.

The database of the test instances described for the subproblem considering Regulation (EC) No 561/2006 was enriched by adding information about gas stations along the route, based on real data. To obtain driving durations and distances between locations and for detours to gas stations, a modified $A^*$ routing algorithm was used. Vehicle consumption rates, tank capacities and start and end fuel levels were taken from the real data from the one-week trips considered. The locations of gas stations considered all over Europe were provided by the service station chain the partner haulage company had fuel cards for. Fuel prices at gas stations were assumed to be equal to the list prices provided by the fuel card operator for the corresponding countries.

From the experience made before we anticipated significantly longer run times if many locations were involved. To reduce the run times for the solution process, a preprocessing heuristic was developed to eliminate unattractive gas stations and thus reduce the solution space. The filter distance was the control parameter which was varied in our numerical experiments to obtain different numbers of remaining gas stations per instance. The larger the filter distance, the more gas stations were eliminated. Additionally, time limits were set up for the different optimization steps to restrict the overall run time to 30 minutes. Here, we made the exception that in each optimization step at least a feasible solution had to be found before stopping the solution process.

Numerical experiments were conducted for the 225 test instances with the extended data described above. For each of the instances, test runs were performed for the filter distances of 100, 200, 300, 400, 500 and 1000 km to analyze the influence of the filter distance on the run time. As expected, run times, especially when considering worst case scenarios, strongly increased with decreasing filter distance and thus with an increasing number of locations considered. On the other hand, the analysis of the solution quality showed that there are only slight improvements in lateness, completion time and fuel expenditures when considering more gas stations. The test results suggest that it is legitimate to choose a rather large filter distance. Using a filter distance of 100 km does not seem to be reasonable at all as the run time on average was 233.47 seconds with 12 instances that had a run time of more than 25 minutes. Of these 12 instances, 11 (4.89\%) were not solved to optimality, that means only a feasible solution was obtained in at least one of the 4 optimization steps. With a filter distance of 1000 km all instances were solved optimally in reasonable times. On average, here the solution process took 11.35 seconds. When considering a filter distance of 500 km, the solution process took on average 32.71 seconds (288.19\%) and for two cases (0.89\%) the run time was more than 25 minutes and no optimal solution was found. For a filter distance of 1000 km, the overall lateness was only 0.08\% worse than the best overall lateness, which amounts to an average difference of only 9 seconds per instance. On average, the fuel costs were only 0.04\% higher than those obtained for the filter distance of 200 km with the lowest fuel cost, that means 38 ct per test instance. The average completion time is 0.03\% later which corresponds to only 2.11 minutes more in the mean. Thus, among the filter distances considered the filter distance of 1000 km for our test setting seems to be the most reasonable one.

Together with the preprocessing heuristic and the right choice of the filter distance good solutions within a reasonable time were obtained for the test instances provided. As test
instances were drawn from real-world data this is a good result. In this thesis, we con-
centrated on long-haul requests. In other settings, more customer locations may have to 
be considered. This increases the problem complexity and thus the computing duration. 
Therefore, some ideas were presented to develop a heuristic for planning driver rest periods 
and vehicle refueling with the same restrictions considered in the MILP model.

5.2. Future research

The basis for the planning of vehicle routes should be reliable driving durations that con-
sider various traffic conditions that are dependent on the routes traveled and the time 
of the day (see, for example, Kok (2010)). Travel times may vary significantly as there 
are differences, for example, between traveling on a Saturday, in rush-hour traffic or at 
the start of vacations. Even though online re-planning is possible, when scheduling driver 
activities, the estimated driving duration should consider time buffers to compensate for 
delays due to unexpected events such as traffic jams or detours because of blocked roads. 
To find reasonable time buffers when scheduling customer stops and driver activities is 
worth further consideration. The robustness depending on the measures chosen could be 
analyzed by simulating deteriorations and using online re-planning with a lesser degree of 
freedom to determine modified schedules.

The presented techniques to plan driver activities do not consider the location of possible 
rest areas. Especially when a daily rest period has to be taken, drivers often face the 
problem of finding an adequate location. Even though modern parking guidance systems 
are available at some locations, nowadays rest areas are often overcrowded. Depending on 
the time of the day, drivers often have to search intensively for a place to spend their daily 
rest period. Goel (2012) proposes an approach that only allows breaks and rest periods 
at rest areas. It is worth taking a closer look at the integration of information about rest 
areas into the model with multiple soft time windows. Additionally, detours to reach rest 
areas could also be considered.

of the European Union (2002)) should be studied. This comprises the rule that working 
time has to be interrupted by a break of at least 30 minutes if the sum of all working hours 
is between six and nine hours and of at least 45 minutes if the driver works more than 
nine hours. Furthermore, Directive 2002/15/EC contains a framework to define rules for 
night work. If night work is performed, the daily working time is not allowed to exceed 
ten hours in each 24 hours period and it has to be compensated by the employer. Different 
implementations in national laws exist. The numerical experiments in Section 2.9.3 show 
that a significant part of the overall working time can be at night if no measures are taken 
to keep it low. Depending on the strategy of a haulage company it would, for example, 
be possible to define time intervals in which no work is allowed. Another approach would 
be to allow not more than 10 hours working time between two daily rest periods and to 
introduce a cost function for working time at night.

Especially when considering long-haul trips, driving bans on public holidays need to be 
integrated into driver scheduling. As there are regional differences, the integration into
the combined problem of route planning (or vehicle routing) and driver scheduling seems reasonable.

Truck drivers face difficult working conditions. In long-haul international transport, truck drivers spend long periods on-road away from home. Competitors and client demands such as just-in-time management induce high pressure. Moreover, remote monitoring and complex technology act as a deterrent because drivers may feel permanently observed and monitored or demoralized. Accessibility of facilities and services (hygienic, food and medical) is not always the best and road safety risks are not negligible. All these reasons lead to a low attractiveness of the profession and a shortage of qualified drivers (European Commission (2014)). Besides lowering pressure by employing better and more realistic planning techniques, an improvement on working conditions can be achieved. Planning techniques that take into account different amenities at resting places where drivers take their daily rest periods and breaks would help to raise the attractiveness of the profession.

In our numerical experiments, prices at gas stations were considered to be constant per country. This approach may be reasonable if only a specific group of gas stations with identical prices per country is considered to be suitable for the refueling of trucks and the driver needs. This in turn may depend on existing contracts with fuel card operators. In Section 2.9.3 we noticed that the overall detour distance on average was best for a filter distance of 1000 km. As described there, we assume that this may be due to the consideration of country prices. It would also be interesting to analyze if the number of refueling stops would change when considering different price structures. Therefore, the analysis of the effect of varying prices within countries would be interesting.

In our mathematical experiments, end fuel levels were taken from real data and thus were input data. In reality, end fuel levels have to be determined prior to the start of the solution process. They should orientate on the fuel price trend and on the price structure of countries expected to be visited in the future. The determination of a good end fuel level can be a future field of research.

Fuel consumption rates depend on the road type, geographical properties of the road, vehicle characteristics, and the driving behavior. Thus, it would be interesting to incorporate additional information, for example, from a geographic information system, from the standardized fleet management system interface of the vehicle and other historical data when determining fuel consumptions between locations for the MILP model input.

The development of a heuristic solution process for the combined problem can be attractive for smaller companies that do not want to use a commercial optimization solver. This also holds if the problem sizes are larger. Some ideas were presented to develop a heuristic according to a metaheuristic scheme. The choice of a metaheuristic and the specific implementation is left for future research.
Appendices
A. Parameters and variables of the combined MILP model

In this appendix a complete overview of all parameters (Appendix A.1) and variables (Appendix A.2) of the MILP model for the combined planning of time windows, rest periods and breaks and vehicle refueling is given.

A.1. Parameters of the MILP model

$r \in \mathbb{N}$  
Total number of vertices representing origin (vertex 0) and destination (vertex $r - 1$), customer locations and gas stations. The vertices are numbered from 0 to $r - 1$ according to the sequence of customer locations to be visited and gas stations that are passed.

$S_{locations}$  
Set of all vertices including all vertices for customer locations, gas stations, origin and destination

$S_{customers} \subset S_{locations}$  
Set of all vertices that correspond to customer locations

$S_{stations} \subset S_{locations}$  
Set of all vertices that correspond to gas stations

$f_{\text{start}} \in \mathbb{N}_0$  
Amount of fuel in the tank at start location 0 in liters

$f_{\text{end}} \in \mathbb{N}_0$  
Minimum amount of fuel to be left in the tank at the final destination $r - 1$ in liters

$P_i \in \mathbb{R}_0^+$  
Fuel price at gas station $i \in S_{stations}$ in € per liter

$\Delta_{\text{min}} \in \mathbb{R}_0^+$  
Minimum amount of fuel to purchase at a gas station in liters

$T_{\text{max}} \in \mathbb{R}_0^+$  
Vehicle tank capacity in liters

$T_{\text{min}} \in \mathbb{R}_0^+$  
Lower bound fuel, i.e. the minimum amount of fuel to be maintained in the tank at all times in liters
\(\Delta_{dr}^{i,i+1} \in \mathbb{N}_0\) \(\Delta_{dr}^{i,i+1}\) Driving time in minutes needed to travel from \(i\) to \(i + 1\), \(i = 0, \ldots, r - 2\) not including the time needed for out of route distances to and from gas stations

\(\Delta_{iTo} \in \mathbb{N}_0\) \(\Delta_{iTo}\) Driving time in minutes needed to travel from the point of departure to the corresponding gas station \(i\) (equals 0 if \(i \notin S_{\text{stations}}\))

\(\Delta_{iFrom} \in \mathbb{N}_0\) \(\Delta_{iFrom}\) Driving time in minutes needed to travel from the gas station \(i\) to the corresponding point of return (equals 0 if \(i \notin S_{\text{stations}}\))

\(\Delta_{cons}^{i,i+1} \in \mathbb{R}_0^+\) \(\Delta_{cons}^{i,i+1}\) Fuel consumption in liters when traveling from \(i\) to \(i + 1\), \(i = 0, \ldots, r - 2\) not including the consumption for out of route distances to and from gas stations

\(\Delta_{consTo} \in \mathbb{R}_0^+\) \(\Delta_{consTo}\) Fuel consumption in liters when traveling from the point of departure to the corresponding gas station \(i\) (equals 0 if \(i \notin S_{\text{stations}}\))

\(\Delta_{consFrom} \in \mathbb{R}_0^+\) \(\Delta_{consFrom}\) Fuel consumption in liters needed to travel from the gas station \(i\) to the corresponding point of return (equals 0 if \(i \notin S_{\text{stations}}\))

\(\Delta_{\text{refuel}} \in \mathbb{N}_0\) \(\Delta_{\text{refuel}}\) Time needed for refueling in minutes

\(\Delta_{\text{service}} \in \mathbb{N}_0\) \(\Delta_{\text{service}}\) Time needed for loading and/or unloading at vertex \(i\), \(i \in S_{\text{customers}}\), in minutes, \(\Delta_{\text{service}} = 0\) and \(\Delta_{\text{service}} = 0\) \(\forall i \in S_{\text{stations}}\)

\(noTW_i \in \mathbb{N}_0\) \(noTW_i\) Number of time windows at customer location \(i\), \(i \in S_{\text{customers}}\)

\(TW_{iz}^{\text{begin}} \in \mathbb{N}_0\) \(TW_{iz}^{\text{begin}}\) Lower limit of the time window \(z\) at vertex \(i\), \(i \in S_{\text{customers}}\), \(z = 0, \ldots, noTW_i - 1\) in minutes counted from start time 0

\(TW_{iz}^{\text{end}} \in \mathbb{N}_0\) \(TW_{iz}^{\text{end}}\) Upper limit of the time window \(z\) at vertex \(i\), \(i \in S_{\text{customers}}\), \(z = 0, \ldots, noTW_i - 1\) in minutes counted from start time 0

\(\overline{udt} \in \mathbb{N}_0\) \(\overline{udt}\) Driving time since the last daily rest period or break at the beginning of the planning horizon in minutes

\(\overline{ddt} \in \mathbb{N}_0\) \(\overline{ddt}\) Cumulated daily driving time since the end of the last daily rest period at the beginning of the planning horizon in minutes

\(\overline{ptr} \in \mathbb{N}_0\) \(\overline{ptr}\) Passed time since the end of the last daily rest period at the
A.1. Parameters of the MILP model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ptwr \in \mathbb{N}_0$</td>
<td>Passed time since the end of the last weekly rest period at the beginning of the planning horizon in minutes</td>
</tr>
<tr>
<td>$urt \in \mathbb{N}_0$</td>
<td>If a daily rest period takes place at start time, this parameter expresses its duration since the start of the rest period in minutes</td>
</tr>
<tr>
<td>$ubt \in \mathbb{N}_0$</td>
<td>If a break takes place at start time, this parameter expresses its duration since the start of the break in minutes</td>
</tr>
<tr>
<td>$dte \in {0,1}$</td>
<td>Is equal to 1 if a driving time extension is currently used when the planning horizon begins, 0 otherwise</td>
</tr>
<tr>
<td>$hpb \in {0,1}$</td>
<td>Is equal to 1 if the first part of a break with a duration of at least 15 minutes has already been taken before the beginning of the planning horizon, 0 otherwise</td>
</tr>
<tr>
<td>$hpr \in {0,1}$</td>
<td>Is equal to 1 if the first part of a daily rest period with a duration of at least 3 hours has already been taken before the beginning of the planning horizon, 0 otherwise</td>
</tr>
<tr>
<td>$noRed \in {0,1,2,3}$</td>
<td>The number of reduced daily rest periods that have already been taken in the current week</td>
</tr>
<tr>
<td>$noExt \in {0,1,2}$</td>
<td>The number of extended daily driving times that have already been taken in the current week</td>
</tr>
</tbody>
</table>
A.2. Variables of the MILP model

Variables needed to define the first two objective functions

\[ \text{start}_i \in \mathbb{R}_0^+ \]  
Start of loading and/or unloading if vertex \( i \in S_{\text{customers}} \),
start of refueling if \( i \in S_{\text{stations}} \),
start of driving (after a potential break or rest period) if \( i = 0 \)

\[ \Delta_{i}^{\text{late}} \in \mathbb{R}_0^+ \]  
Lateness in vertex \( i, i = 1, \ldots, r - 1 \). Is set to be zero if the considered vertex does not correspond to a customer location.

\[ \Delta_{i}^{\text{refuel}} \in \mathbb{R}_0^+ \]  
Amount of fuel to purchase at gas station \( i \in S_{\text{stations}} \) in liters

Variables for the integration of refueling decisions including those that have been modified for the integration

\[ \alpha_{i}^{\text{refuel}} = \begin{cases} 1 & \text{if } i \in S_{\text{stations}} \text{ and } i \text{ is selected for refueling} \\ 0 & \text{otherwise} \end{cases} \]

\[ T_i \in \mathbb{R}_0^+ \]  
Amount of fuel in the tank either at truck stop \( i \) before purchasing fuel (\( \Delta_{i}^{\text{refuel}} = 1 \)) or at the corresponding leaving point (\( \Delta_{i}^{\text{refuel}} = 0 \))

\[ \Delta_{(i,i+1)}^{\text{dr}} \in \mathbb{R}_0^+ \]  
Driving duration between locations. If \( i \) is a gas station and refueling takes place at \( i \), \( \Delta_{(i,i+1)}^{\text{dr}} \) includes the out-of-route driving duration from gas station \( i \). If refueling takes place at \( i + 1 \), the out-of-route driving duration to gas station \( i + 1 \) is added.

\[ \Delta_{i}^{\text{work}} \in \mathbb{R}_0^+ \]  
Time needed for loading and/or unloading at location \( i \) in minutes if \( i \) is associated with a gas station, i.e. \( i \in S_{\text{customers}} \).
Time needed for refueling if \( i \in S_{\text{stations}} \). \( \Delta_{i}^{\text{work}} \) is set to be 0 if \( i \in S_{\text{stations}} \) and no refueling takes place in \( i \).
A.2. Variables of the MILP model

Variables that indicate which time window is chosen at customer $i$

$$tw_{iz} = \begin{cases} 
1 & \text{if time window } z \text{ is chosen at destination } i \in S^{customers} \\
0 & \text{otherwise} 
\end{cases}$$

$i = 1, \ldots, r - 1, z = 0, \ldots, nbTW_i - 1$

The following set comprises the continuous status variables for each vertex $i$.

$E_{dt}^i$ Driving time left until the next break or daily rest period when entering vertex $i$, $i = 0, \ldots, r - 1$ in minutes
$$0 \leq E_{dt}^i \leq 270$$

$L_{dt}^i$ Driving time left until the next break or daily rest period when leaving vertex $i$, $i = 0, \ldots, r - 1$ in minutes
$$0 \leq L_{dt}^i \leq 270$$

$E_{ddt}^i$ Driving time left until the next daily period rest when entering vertex $i$
$$i = 0, \ldots, r - 1 \text{ in minutes}$$
$$0 \leq E_{ddt}^i \leq 540$$

$L_{ddt}^i$ Driving time left until the next daily period rest when leaving vertex $i$
$$i = 0, \ldots, r - 1 \text{ in minutes}$$
$$0 \leq L_{ddt}^i \leq 540$$

$E_{t}^i$ Time left until the next daily rest period when entering vertex $i$
$$i = 0, \ldots, r - 1 \text{ in minutes}$$
$$0 \leq E_{t}^i \leq 900$$

$L_{t}^i$ Time left until the next daily rest period when leaving vertex $i$
$$i = 0, \ldots, r - 1 \text{ in minutes}$$
$$0 \leq L_{t}^i \leq 900$$
The following variables indicate for each arc \((i, i+1)\) if a daily rest period is taken, the number of daily rest periods and their cumulated duration.

\[
\alpha_{i(i+1)}^{rest} = \begin{cases} 
1 & \text{if at least one daily rest period is taken on arc } (i, i+1) \\
0 & \text{otherwise}
\end{cases}
\]

\[i = 0, \ldots, r-2\]

\[\mathcal{A}_{i(i+1)}^{rest} \in \mathbb{N}_0\] The number of daily rest periods taken on arc \((i, i+1)\),

\[i = 0, \ldots, r-2\]

\[\Delta_{i(i+1)}^{rest} \in \mathbb{R}^+_0\] The cumulated duration of all daily rest periods on arc \((i, i+1)\),

\[i = 0, \ldots, r-2\]

Regarding daily rest periods at vertices, the following variables indicate if a daily rest period is made and its duration.

\[
\alpha_i^{rest} = \begin{cases} 
1 & \text{if a daily rest period is made in vertex } i \\
0 & \text{otherwise}
\end{cases}
\]

\[i = 0, \ldots, r-1\]

\[\Delta_i^{rest} \in \mathbb{R}^+_0\] The duration of a daily rest period in vertex \(i\),

\[i = 0, \ldots, r-1\]

The next set of variables are needed to determine if breaks are taken on arc \((i, i+1)\) and their number.

\[
\alpha_{i(i+1)}^{break} = \begin{cases} 
1 & \text{if at least one break is taken on arc } (i, i+1) \\
0 & \text{otherwise}
\end{cases}
\]

\[i = 0, \ldots, r-2\]

\[\mathcal{A}_{i(i+1)}^{break} \in \mathbb{N}_0\] The number of breaks taken on arc \((i, i+1)\), \(i = 0, \ldots, r-2\)
The following variables indicate if breaks are taken in vertices.

\[ \alpha_i^{\text{break}} = \begin{cases} 
1 & \text{if a break is taken in vertex } i \\
0 & \text{otherwise} 
\end{cases} \]

\[ i = 0, \ldots, r - 1 \]

Each variable \( \Delta_i^{\text{wait}} \) gives the waiting time in vertex \( i \):

\[ \Delta_i^{\text{wait}} \in \mathbb{R}_0^+ \quad \text{Waiting time in vertex } i, \ i = 0, \ldots, r - 1 \]

The next variables specify if an early daily rest period is taken on an arc, meaning that the daily driving time is not completely used up.

\[ \mu_{(i,i+1)}^{\text{earlydr1}} = \begin{cases} 
1 & \text{if a break is replaced by a daily rest period on arc } (i, i+1) \\
0 & \text{and this rest is the first rest on this arc} 
\end{cases} \]

\[ i = 0, \ldots, r - 2 \]

\[ \mu_{(i,i+1)}^{\text{earlydr2}} = \begin{cases} 
1 & \text{if a break is replaced by a daily rest period on arc } (i, i+1) \\
0 & \text{and this rest is not the first rest on this arc} 
\end{cases} \]

\[ i = 0, \ldots, r - 2 \]

When arriving in vertex \( i \), in case a daily rest period was taken on arc \( (i-1,i) \), the following variable indicates if a break was taken since the last daily rest period.

\[ e_i^{\text{bt}} = \begin{cases} 
1 & \text{if the last rest activity on the preceding arc } (i-1,i) \text{ was a break} \\
0 & \text{otherwise} 
\end{cases} \]

\[ i = 0, \ldots, r - 1 \]
The next variables indicate if a break is still necessary to completely use up the daily driving time left when leaving vertex $i$.

$$l^b_i = \begin{cases} 
1 & \text{if a break would be necessary to completely exploit} \\
0 & \text{the daily driving time left when leaving vertex } i
\end{cases}$$

$i = 0, \ldots, r - 1$

The following variables are needed to model the optional rules.

$$\alpha^\text{break}_i = \begin{cases} 
1 & \text{if the first part of a break is taken in vertex } i \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$

$$\mu^\text{upbreak}_{(i,i+1)} = \begin{cases} 
1 & \text{if the second part of a break is taken on arc } (i, i + 1) \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 2$

$$\mu^\text{upbreak}_i = \begin{cases} 
1 & \text{if the second part of a break is taken in vertex } i \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$

$$l^\text{break}_i = \begin{cases} 
1 & \text{if when leaving vertex } i \text{ a partial break of } 15 \text{ minutes was taken} \\
0 & \text{since the last rest period}
\end{cases}$$

$i = 0, \ldots, r - 1$

$$\alpha^\text{rest}_i = \begin{cases} 
1 & \text{if the first part of a daily rest period is taken in vertex } i \\
0 & \text{otherwise}
\end{cases}$$

$i = 0, \ldots, r - 1$
A.2. Variables of the MILP model

\begin{align*}
\mu_{i}^{\text{prest}} &= \begin{cases} 
1 & \text{if when leaving vertex } i \text{ a partial rest period of 3 h was taken since the last rest period} \\
0 & \text{otherwise}
\end{cases} \\
i &= 0, \ldots, r - 1
\end{align*}

\begin{align*}
\mu_{i}^{\text{dredrest}} &= \begin{cases} 
1 & \text{if the last break on arc } (i - 1, i) \text{ is substituted by a first partial daily rest period} \\
0 & \text{otherwise}
\end{cases} \\
i &= 1, \ldots, r - 1
\end{align*}

\begin{align*}
\mu_{(i, i+1)}^{\text{redrest}} &\in \{0, 1, 2, 3\} \quad \text{The number of reduced daily rest periods taken on arc } (i, i + 1), \ i = 0, \ldots, r - 2
\end{align*}

\begin{align*}
\mu_{i}^{\text{redrest}} &= \begin{cases} 
1 & \text{if a reduced daily rest period is taken in vertex } i \\
0 & \text{otherwise}
\end{cases} \\
i &= 0, \ldots, r - 1
\end{align*}

\begin{align*}
\mu_{i}^{\text{dredrest}} &= \begin{cases} 
1 & \text{if the next daily rest period is a reduced one and is taken after leaving vertex } i \\
0 & \text{otherwise}
\end{cases} \\
i &= 0, \ldots, r - 1
\end{align*}

\begin{align*}
\mu_{(i, i+1)}^{\text{extd}} &= \begin{cases} 
1 & \text{if a driving time extension is used on arc } (i, i + 1) \text{ before the first daily rest period} \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\[ i = 0, \ldots, r - 2 \]

\[ \mu_{(i,i+1)}^{\text{extd2}} \in \{0, 1, 2\} \quad \text{The number of driving time extensions used on arc } (i, i+1) \]
\[ \text{between the first and the last daily rest period, } i = 0, \ldots, r - 2 \]

\[ \mu_{(i,i+1)}^{\text{extd3}} = \begin{cases} 
    1 & \text{if a driving time extension is used on arc } (i, i+1) \text{ after the} \\
    0 & \text{last daily rest period} \\
\end{cases} 
\]

\[ i = 0, \ldots, r - 2 \]

\[ \mu_i^{\text{extd}} = \begin{cases} 
    1 & \text{if a driving time extension is decided in vertex } i \\
    0 & \text{otherwise} \\
\end{cases} 
\]

\[ i = 0, \ldots, r - 1 \]

\[ \nu_i^{\text{extd}} = \begin{cases} 
    1 & \text{if a decision concerning a driving time extension was made} \\
    0 & \text{before leaving vertex } i \\
\end{cases} 
\]

\[ i = 0, \ldots, r - 1 \]

**Auxiliary variables:**

\[ \lambda_i^1, \lambda_i^2, \lambda_i^3, \lambda_i^4, \lambda_i^5, \lambda_i^6, \lambda_i^7 \in \{0, 1\}, \quad i = 0, \ldots, r - 1 \]

\[ \lambda_i^5 \in \{0, 1\}, \quad i = 0, \ldots, r - 2 \]
B. Detailed results of numerical experiments for the combined problem

This appendix presents detailed results of the mathematical experiments conducted for the combined problem by showing the overall run times for all test runs. The next six tables depict the run times for all 225 test instances for the different filter distances chosen for the preprocessing heuristic. In each table, the instances are categorized according to the number and length of time windows considered. Additionally, the number of customer locations and the overall number of locations is given on the left-hand side.
<table>
<thead>
<tr>
<th>base</th>
<th># loc.</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>inst.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>36.85</td>
<td>48.56</td>
<td>35.62</td>
<td>25.30</td>
<td>33.31</td>
<td>179.64</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>101.46</td>
<td>113.79</td>
<td>211.16</td>
<td>114.51</td>
<td>97.31</td>
<td>638.23</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>603.63</td>
<td>901.81</td>
<td>575.84</td>
<td>291.52</td>
<td>475.21</td>
<td>2848.02</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>4.45</td>
<td>5.18</td>
<td>16.86</td>
<td>5.63</td>
<td>4.87</td>
<td>36.98</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>11.48</td>
<td>11.75</td>
<td>9.05</td>
<td>6.29</td>
<td>45.69</td>
<td>56.06</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>16.12</td>
<td>17.93</td>
<td>5.21</td>
<td>14.46</td>
<td>38.07</td>
<td>91.78</td>
</tr>
<tr>
<td>7</td>
<td>29</td>
<td>34.29</td>
<td>21.85</td>
<td>12.00</td>
<td>19.28</td>
<td>19.45</td>
<td>106.87</td>
</tr>
<tr>
<td>8</td>
<td>39</td>
<td>127.13</td>
<td>83.26</td>
<td>92.74</td>
<td>145.24</td>
<td>75.10</td>
<td>523.46</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
<td>21.34</td>
<td>45.23</td>
<td>96.40</td>
<td>35.16</td>
<td>57.02</td>
<td>255.14</td>
</tr>
<tr>
<td>10</td>
<td>33</td>
<td>258.87</td>
<td>61.04</td>
<td>110.33</td>
<td>169.40</td>
<td>154.22</td>
<td>753.86</td>
</tr>
<tr>
<td>11</td>
<td>35</td>
<td>100.20</td>
<td>39.28</td>
<td>38.19</td>
<td>51.26</td>
<td>58.30</td>
<td>287.23</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>35.23</td>
<td>44.84</td>
<td>42.28</td>
<td>86.55</td>
<td>82.42</td>
<td>291.30</td>
</tr>
<tr>
<td>13</td>
<td>30</td>
<td>97.63</td>
<td>64.69</td>
<td>22.92</td>
<td>51.31</td>
<td>69.23</td>
<td>305.78</td>
</tr>
<tr>
<td>14</td>
<td>38</td>
<td>135.58</td>
<td>232.05</td>
<td>111.49</td>
<td>201.99</td>
<td>937.24</td>
<td>1618.35</td>
</tr>
<tr>
<td>15</td>
<td>42</td>
<td>293.27</td>
<td>1511.20</td>
<td>1574.46</td>
<td>1514.27</td>
<td>479.55</td>
<td>5372.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># time windows</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1877.50</td>
<td>3202.45</td>
<td>2954.54</td>
<td>2732.17</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>26</td>
<td>42.85</td>
<td>29.19</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>29</td>
<td>104.71</td>
<td>323.67</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>32</td>
<td>128.20</td>
<td>98.31</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>29</td>
<td>8.27</td>
<td>18.33</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>21</td>
<td>4.68</td>
<td>6.43</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>30</td>
<td>18.36</td>
<td>93.13</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>29</td>
<td>56.60</td>
<td>14.39</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>39</td>
<td>133.68</td>
<td>131.14</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>32</td>
<td>11.47</td>
<td>28.41</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>33</td>
<td>127.47</td>
<td>235.97</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>35</td>
<td>159.59</td>
<td>71.08</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>32</td>
<td>126.10</td>
<td>81.12</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>30</td>
<td>49.03</td>
<td>39.52</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>38</td>
<td>221.40</td>
<td>216.90</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>42</td>
<td>614.10</td>
<td>201.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Σ</th>
<th>1823.30</th>
<th>1588.61</th>
<th>2453.64</th>
<th>4210.40</th>
<th>4078.33</th>
<th>14154.28</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>26</td>
<td>33.92</td>
<td>53.51</td>
<td>29.97</td>
<td>45.77</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>29</td>
<td>209.12</td>
<td>258.18</td>
<td>31.34</td>
<td>190.77</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>32</td>
<td>223.66</td>
<td>85.88</td>
<td>121.41</td>
<td>263.19</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>29</td>
<td>12.06</td>
<td>9.33</td>
<td>6.29</td>
<td>12.76</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>21</td>
<td>14.82</td>
<td>12.17</td>
<td>7.52</td>
<td>11.40</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>30</td>
<td>24.93</td>
<td>50.92</td>
<td>40.84</td>
<td>35.62</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>29</td>
<td>47.11</td>
<td>16.13</td>
<td>228.62</td>
<td>13.93</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>39</td>
<td>286.98</td>
<td>145.00</td>
<td>235.83</td>
<td>261.43</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>32</td>
<td>47.08</td>
<td>29.78</td>
<td>37.46</td>
<td>56.92</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>33</td>
<td>141.67</td>
<td>267.84</td>
<td>505.21</td>
<td>271.08</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>35</td>
<td>983.09</td>
<td>75.36</td>
<td>101.64</td>
<td>146.22</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>32</td>
<td>93.10</td>
<td>141.87</td>
<td>121.13</td>
<td>149.30</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>30</td>
<td>67.11</td>
<td>231.57</td>
<td>68.16</td>
<td>128.12</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>38</td>
<td>440.24</td>
<td>244.58</td>
<td>881.94</td>
<td>3508.53</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>42</td>
<td>1576.56</td>
<td>1546.64</td>
<td>219.87</td>
<td>1151.88</td>
</tr>
</tbody>
</table>

| Σ              | 4201.44 | 3168.75 | 2637.21 | 6192.91 | 8821.30 | 25021.61 |

Table B.1.: Filter distance 100 km: Run times in seconds for the MILP model solution process
<table>
<thead>
<tr>
<th>base inst.</th>
<th># loc.</th>
<th>length of time windows in min</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>2.43</td>
<td>2.26</td>
<td>3.12</td>
<td>2.75</td>
<td>2.98</td>
<td>13.54</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>9.92</td>
<td>8.57</td>
<td>7.35</td>
<td>4.57</td>
<td>4.48</td>
<td>34.86</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>64.09</td>
<td>114.75</td>
<td>77.48</td>
<td>19.75</td>
<td>23.24</td>
<td>299.32</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>2.51</td>
<td>2.45</td>
<td>1.76</td>
<td>1.64</td>
<td>2.57</td>
<td>10.94</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>3.32</td>
<td>2.48</td>
<td>2.35</td>
<td>2.57</td>
<td>8.68</td>
<td>13.01</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>1.76</td>
<td>1.86</td>
<td>0.94</td>
<td>2.95</td>
<td>6.52</td>
<td>14.02</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>25.30</td>
<td>13.48</td>
<td>7.02</td>
<td>8.32</td>
<td>34.94</td>
<td>89.06</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>12.00</td>
<td>9.84</td>
<td>24.38</td>
<td>12.17</td>
<td>10.20</td>
<td>68.55</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>3.57</td>
<td>6.86</td>
<td>5.06</td>
<td>6.99</td>
<td>9.94</td>
<td>32.42</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>27</td>
<td>51.29</td>
<td>30.33</td>
<td>61.45</td>
<td>59.06</td>
<td>67.07</td>
<td>269.20</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>26</td>
<td>31.14</td>
<td>38.10</td>
<td>9.16</td>
<td>39.39</td>
<td>55.13</td>
<td>172.91</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>20.72</td>
<td>10.34</td>
<td>20.52</td>
<td>19.11</td>
<td>17.85</td>
<td>88.53</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>23</td>
<td>53.31</td>
<td>32.71</td>
<td>11.84</td>
<td>27.91</td>
<td>20.87</td>
<td>146.64</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>29</td>
<td>23.76</td>
<td>36.08</td>
<td>52.32</td>
<td>48.16</td>
<td>45.79</td>
<td>206.11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>34</td>
<td>50.76</td>
<td>1508.83</td>
<td>264.31</td>
<td>597.50</td>
<td>1582.46</td>
<td>4003.86</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>355.89</td>
<td>1818.94</td>
<td>549.06</td>
<td>852.38</td>
<td>1886.72</td>
<td>5462.98</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>2.50</td>
<td>2.20</td>
<td>2.79</td>
<td>3.43</td>
<td>3.12</td>
<td>14.04</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>3.14</td>
<td>3.29</td>
<td>3.57</td>
<td>3.20</td>
<td>3.85</td>
<td>17.05</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>17.75</td>
<td>36.04</td>
<td>36.30</td>
<td>11.01</td>
<td>23.45</td>
<td>124.55</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>3.34</td>
<td>3.28</td>
<td>1.79</td>
<td>11.22</td>
<td>2.81</td>
<td>22.43</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>2.09</td>
<td>1.47</td>
<td>2.47</td>
<td>2.28</td>
<td>5.24</td>
<td>13.54</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>1.90</td>
<td>3.79</td>
<td>4.18</td>
<td>7.75</td>
<td>9.08</td>
<td>26.71</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>8.05</td>
<td>7.08</td>
<td>4.68</td>
<td>6.26</td>
<td>3.29</td>
<td>29.36</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>26</td>
<td>14.46</td>
<td>14.42</td>
<td>11.69</td>
<td>20.83</td>
<td>32.98</td>
<td>94.37</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>22</td>
<td>5.21</td>
<td>6.80</td>
<td>7.57</td>
<td>8.64</td>
<td>34.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>27</td>
<td>47.05</td>
<td>36.16</td>
<td>41.34</td>
<td>31.75</td>
<td>90.01</td>
<td>246.31</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>26</td>
<td>107.91</td>
<td>58.56</td>
<td>27.71</td>
<td>43.01</td>
<td>27.16</td>
<td>264.34</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>25</td>
<td>39.78</td>
<td>30.64</td>
<td>33.79</td>
<td>19.06</td>
<td>10.00</td>
<td>133.27</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>23</td>
<td>12.46</td>
<td>34.23</td>
<td>17.32</td>
<td>11.73</td>
<td>24.67</td>
<td>100.40</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>29</td>
<td>34.87</td>
<td>34.23</td>
<td>49.42</td>
<td>10.78</td>
<td>82.78</td>
<td>212.07</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>34</td>
<td>82.57</td>
<td>451.55</td>
<td>360.27</td>
<td>1512.62</td>
<td>1544.85</td>
<td>3951.85</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>383.07</td>
<td>723.72</td>
<td>604.88</td>
<td>1701.46</td>
<td>1871.92</td>
<td>5285.04</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>2.72</td>
<td>2.79</td>
<td>2.76</td>
<td>3.45</td>
<td>3.43</td>
<td>15.15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>12.78</td>
<td>5.73</td>
<td>9.55</td>
<td>4.79</td>
<td>29.59</td>
<td>62.43</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>24.21</td>
<td>34.53</td>
<td>22.65</td>
<td>17.11</td>
<td>37.92</td>
<td>136.42</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>3.60</td>
<td>4.21</td>
<td>3.67</td>
<td>4.68</td>
<td>3.37</td>
<td>19.53</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>4.52</td>
<td>2.60</td>
<td>2.60</td>
<td>3.93</td>
<td>9.52</td>
<td>23.18</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>4.40</td>
<td>8.19</td>
<td>7.33</td>
<td>5.40</td>
<td>15.26</td>
<td>40.58</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>7.58</td>
<td>12.26</td>
<td>9.00</td>
<td>12.82</td>
<td>32.15</td>
<td>73.82</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>26</td>
<td>24.38</td>
<td>42.96</td>
<td>30.65</td>
<td>16.66</td>
<td>58.95</td>
<td>173.61</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>3.51</td>
<td>41.57</td>
<td>5.24</td>
<td>9.06</td>
<td>12.15</td>
<td>71.54</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>27</td>
<td>104.60</td>
<td>111.65</td>
<td>82.42</td>
<td>87.98</td>
<td>263.80</td>
<td>650.45</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>26</td>
<td>111.74</td>
<td>41.82</td>
<td>28.22</td>
<td>23.37</td>
<td>56.18</td>
<td>261.33</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>25</td>
<td>23.54</td>
<td>26.46</td>
<td>60.44</td>
<td>32.89</td>
<td>15.35</td>
<td>158.67</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>23</td>
<td>16.96</td>
<td>19.53</td>
<td>47.55</td>
<td>16.88</td>
<td>52.26</td>
<td>153.18</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>29</td>
<td>100.32</td>
<td>87.25</td>
<td>83.84</td>
<td>61.93</td>
<td>241.36</td>
<td>574.71</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>34</td>
<td>482.48</td>
<td>99.59</td>
<td>412.98</td>
<td>182.97</td>
<td>489.19</td>
<td>1667.21</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>927.35</td>
<td>541.16</td>
<td>808.89</td>
<td>483.93</td>
<td>1320.48</td>
<td>4081.81</td>
<td></td>
</tr>
</tbody>
</table>

Table B.2.: Filter distance 200 km: Run times in seconds for the MILP model solution process
### Table B.3: Filter distance 300 km: Run times in seconds for the MILP model solution process

<table>
<thead>
<tr>
<th>base inst.</th>
<th># loc.</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2.39</td>
<td>3.01</td>
<td>3.01</td>
<td>1.42</td>
<td>1.78</td>
<td>11.61</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10.27</td>
<td>10.55</td>
<td>11.62</td>
<td>3.98</td>
<td>10.31</td>
<td>46.72</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.70</td>
<td>2.03</td>
<td>2.34</td>
<td>3.07</td>
<td>2.76</td>
<td>12.90</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.23</td>
<td>2.53</td>
<td>1.31</td>
<td>1.96</td>
<td>3.84</td>
<td>11.46</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.87</td>
<td>1.90</td>
<td>1.47</td>
<td>2.17</td>
<td>6.69</td>
<td>14.10</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>8.49</td>
<td>4.68</td>
<td>4.09</td>
<td>3.37</td>
<td>3.28</td>
<td>23.90</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>6.02</td>
<td>6.85</td>
<td>9.70</td>
<td>4.76</td>
<td>5.71</td>
<td>33.04</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1.95</td>
<td>3.74</td>
<td>2.64</td>
<td>2.39</td>
<td>2.09</td>
<td>12.80</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>24.46</td>
<td>23.60</td>
<td>26.30</td>
<td>26.55</td>
<td>29.22</td>
<td>129.14</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10.61</td>
<td>12.31</td>
<td>10.75</td>
<td>8.89</td>
<td>24.20</td>
<td>66.75</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>7.58</td>
<td>8.86</td>
<td>3.46</td>
<td>10.17</td>
<td>12.18</td>
<td>42.26</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>13.14</td>
<td>14.46</td>
<td>18.08</td>
<td>8.75</td>
<td>11.59</td>
<td>66.02</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>37.78</td>
<td>49.95</td>
<td>38.07</td>
<td>33.38</td>
<td>52.89</td>
<td>212.07</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>77.49</td>
<td>48.81</td>
<td>1520.62</td>
<td>1509.11</td>
<td>1227.73</td>
<td>4383.75</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>207.64</td>
<td>195.24</td>
<td>1655.58</td>
<td>1621.42</td>
<td>1395.85</td>
<td>5075.73</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.62</td>
<td>1.97</td>
<td>1.69</td>
<td>2.58</td>
<td>2.22</td>
<td>10.07</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.51</td>
<td>1.01</td>
<td>1.06</td>
<td>2.03</td>
<td>3.07</td>
<td>8.69</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.70</td>
<td>3.96</td>
<td>3.24</td>
<td>4.84</td>
<td>8.75</td>
<td>23.49</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.04</td>
<td>2.28</td>
<td>2.06</td>
<td>1.76</td>
<td>1.78</td>
<td>9.92</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.83</td>
<td>1.92</td>
<td>2.59</td>
<td>1.86</td>
<td>3.79</td>
<td>11.98</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2.15</td>
<td>3.73</td>
<td>5.94</td>
<td>9.10</td>
<td>6.58</td>
<td>27.50</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4.32</td>
<td>4.87</td>
<td>3.54</td>
<td>6.05</td>
<td>3.53</td>
<td>22.31</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4.60</td>
<td>5.31</td>
<td>7.00</td>
<td>8.56</td>
<td>24.38</td>
<td>49.86</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>2.80</td>
<td>1.84</td>
<td>2.98</td>
<td>2.84</td>
<td>12.16</td>
<td>32.96</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>24.73</td>
<td>37.89</td>
<td>33.63</td>
<td>40.20</td>
<td>36.89</td>
<td>173.35</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>26.05</td>
<td>13.98</td>
<td>11.62</td>
<td>8.36</td>
<td>12.32</td>
<td>72.34</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>22.03</td>
<td>26.33</td>
<td>12.48</td>
<td>32.84</td>
<td>9.41</td>
<td>103.09</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>6.82</td>
<td>6.32</td>
<td>18.60</td>
<td>9.75</td>
<td>17.61</td>
<td>59.10</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>37.53</td>
<td>21.90</td>
<td>21.00</td>
<td>44.02</td>
<td>218.43</td>
<td>342.89</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>1513.61</td>
<td>80.36</td>
<td>252.28</td>
<td>87.53</td>
<td>930.39</td>
<td>2864.18</td>
</tr>
<tr>
<td></td>
<td>Σ</td>
<td>1653.55</td>
<td>213.66</td>
<td>379.71</td>
<td>262.32</td>
<td>1284.00</td>
<td>3793.24</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.04</td>
<td>2.57</td>
<td>2.56</td>
<td>3.17</td>
<td>2.04</td>
<td>12.39</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.62</td>
<td>2.32</td>
<td>2.36</td>
<td>2.31</td>
<td>14.82</td>
<td>25.43</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7.74</td>
<td>4.74</td>
<td>3.79</td>
<td>9.31</td>
<td>13.14</td>
<td>38.72</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.10</td>
<td>3.99</td>
<td>3.25</td>
<td>4.74</td>
<td>3.04</td>
<td>18.13</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2.03</td>
<td>3.64</td>
<td>2.53</td>
<td>2.81</td>
<td>4.43</td>
<td>15.43</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>4.59</td>
<td>6.58</td>
<td>7.38</td>
<td>5.21</td>
<td>9.34</td>
<td>33.10</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>4.06</td>
<td>5.26</td>
<td>6.37</td>
<td>5.71</td>
<td>6.49</td>
<td>27.88</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9.23</td>
<td>16.16</td>
<td>12.28</td>
<td>9.50</td>
<td>20.47</td>
<td>67.64</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>3.64</td>
<td>2.34</td>
<td>2.76</td>
<td>12.79</td>
<td>5.62</td>
<td>27.14</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>45.80</td>
<td>102.34</td>
<td>52.48</td>
<td>86.89</td>
<td>86.99</td>
<td>374.49</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>45.02</td>
<td>21.39</td>
<td>24.60</td>
<td>22.82</td>
<td>23.82</td>
<td>137.66</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>29.77</td>
<td>17.60</td>
<td>25.88</td>
<td>23.74</td>
<td>7.58</td>
<td>104.57</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>35.72</td>
<td>12.14</td>
<td>10.30</td>
<td>13.23</td>
<td>51.98</td>
<td>123.37</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>82.85</td>
<td>50.34</td>
<td>52.90</td>
<td>33.31</td>
<td>117.20</td>
<td>336.60</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>121.84</td>
<td>79.80</td>
<td>109.34</td>
<td>115.35</td>
<td>966.83</td>
<td>1393.15</td>
</tr>
<tr>
<td></td>
<td>Σ</td>
<td>401.05</td>
<td>331.21</td>
<td>318.76</td>
<td>350.89</td>
<td>1333.79</td>
<td>2735.69</td>
</tr>
<tr>
<td># time windows</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Σ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>base inst.</td>
<td># loc.</td>
<td>length of time windows in min</td>
<td>0</td>
<td>30</td>
<td>60</td>
<td>120</td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>2.09</td>
<td>2.59</td>
<td>2.40</td>
<td>2.00</td>
<td>2.34</td>
<td>11.42</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>2.08</td>
<td>2.21</td>
<td>1.76</td>
<td>2.53</td>
<td>11.95</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>5.07</td>
<td>4.85</td>
<td>8.06</td>
<td>3.10</td>
<td>7.61</td>
<td>28.70</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>2.17</td>
<td>1.20</td>
<td>1.06</td>
<td>1.81</td>
<td>2.06</td>
<td>8.30</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>0.91</td>
<td>1.45</td>
<td>1.87</td>
<td>1.12</td>
<td>2.45</td>
<td>17.89</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>1.44</td>
<td>1.19</td>
<td>0.83</td>
<td>1.25</td>
<td>4.35</td>
<td>9.05</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>2.90</td>
<td>3.35</td>
<td>2.92</td>
<td>12.86</td>
<td>3.82</td>
<td>25.85</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>5.23</td>
<td>5.10</td>
<td>5.10</td>
<td>3.67</td>
<td>3.99</td>
<td>23.09</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>2.06</td>
<td>2.04</td>
<td>1.93</td>
<td>2.40</td>
<td>3.28</td>
<td>11.72</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
<td>35.71</td>
<td>17.38</td>
<td>18.24</td>
<td>14.77</td>
<td>31.92</td>
<td>118.02</td>
</tr>
<tr>
<td>11</td>
<td>19</td>
<td>7.60</td>
<td>5.40</td>
<td>7.41</td>
<td>4.80</td>
<td>9.55</td>
<td>34.76</td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>5.86</td>
<td>6.49</td>
<td>7.65</td>
<td>4.45</td>
<td>22.31</td>
<td>46.76</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>11.23</td>
<td>9.45</td>
<td>3.68</td>
<td>3.14</td>
<td>11.84</td>
<td>39.34</td>
</tr>
<tr>
<td>14</td>
<td>25</td>
<td>23.85</td>
<td>18.80</td>
<td>28.45</td>
<td>24.12</td>
<td>20.00</td>
<td>115.22</td>
</tr>
<tr>
<td>15</td>
<td>31</td>
<td>47.00</td>
<td>55.02</td>
<td>133.35</td>
<td>935.40</td>
<td>552.93</td>
<td>1723.70</td>
</tr>
<tr>
<td>Σ</td>
<td>155.19</td>
<td>136.53</td>
<td>226.32</td>
<td>1016.65</td>
<td>680.96</td>
<td>2215.65</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># time windows</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>base inst.</td>
<td># loc.</td>
<td>length of time windows in min</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>1.89</td>
<td>2.17</td>
<td>1.90</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>3.21</td>
<td>1.45</td>
<td>1.58</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>2.28</td>
<td>2.08</td>
<td>0.91</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>2.26</td>
<td>1.89</td>
<td>1.31</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>1.50</td>
<td>1.61</td>
<td>1.37</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>1.08</td>
<td>3.20</td>
<td>3.53</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>3.46</td>
<td>1.95</td>
<td>2.89</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>6.21</td>
<td>4.20</td>
<td>3.87</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>2.76</td>
<td>2.06</td>
<td>2.25</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
<td>49.97</td>
<td>28.52</td>
<td>14.06</td>
</tr>
<tr>
<td>11</td>
<td>19</td>
<td>11.87</td>
<td>8.64</td>
<td>6.90</td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>17.63</td>
<td>8.97</td>
<td>14.43</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>2.54</td>
<td>13.03</td>
<td>16.64</td>
</tr>
<tr>
<td>14</td>
<td>25</td>
<td>35.12</td>
<td>60.06</td>
<td>41.51</td>
</tr>
<tr>
<td>15</td>
<td>31</td>
<td>1521.43</td>
<td>66.47</td>
<td>28.56</td>
</tr>
<tr>
<td>Σ</td>
<td>1663.20</td>
<td>206.28</td>
<td>141.69</td>
<td>277.96</td>
</tr>
</tbody>
</table>

Table B.4.: Filter distance 400 km: Run times in seconds for the MILP model solution process
<table>
<thead>
<tr>
<th>base inst.</th>
<th># loc.</th>
<th>length of time windows in min</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>1.16</td>
<td>1.04</td>
<td>0.77</td>
<td>0.67</td>
<td>0.72</td>
<td>4.35</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>2.34</td>
<td>2.89</td>
<td>2.82</td>
<td>1.54</td>
<td>1.89</td>
<td>11.48</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>2.97</td>
<td>4.56</td>
<td>8.83</td>
<td>4.37</td>
<td>3.73</td>
<td>24.45</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>1.45</td>
<td>0.83</td>
<td>1.22</td>
<td>1.42</td>
<td>1.93</td>
<td>6.85</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>0.64</td>
<td>0.98</td>
<td>0.58</td>
<td>0.84</td>
<td>2.04</td>
<td>5.09</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>1.12</td>
<td>0.90</td>
<td>0.91</td>
<td>1.11</td>
<td>3.93</td>
<td>7.97</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>2.82</td>
<td>3.37</td>
<td>2.86</td>
<td>12.84</td>
<td>3.84</td>
<td>25.73</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>4.42</td>
<td>4.99</td>
<td>4.31</td>
<td>4.54</td>
<td>3.70</td>
<td>21.95</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>1.97</td>
<td>1.78</td>
<td>1.48</td>
<td>1.95</td>
<td>3.10</td>
<td>10.28</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>5.32</td>
<td>19.66</td>
<td>19.92</td>
<td>15.30</td>
<td>27.05</td>
<td>87.25</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>8.11</td>
<td>6.33</td>
<td>3.46</td>
<td>6.15</td>
<td>9.69</td>
<td>33.75</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>5.80</td>
<td>6.55</td>
<td>7.64</td>
<td>4.42</td>
<td>23.90</td>
<td>48.31</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>16.29</td>
<td>13.51</td>
<td>3.70</td>
<td>5.31</td>
<td>13.38</td>
<td>52.18</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>16.10</td>
<td>15.48</td>
<td>26.58</td>
<td>27.82</td>
<td>15.66</td>
<td>101.64</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>25.85</td>
<td>57.30</td>
<td>175.67</td>
<td>391.70</td>
<td>569.29</td>
<td>1219.81</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>96.35</td>
<td>140.17</td>
<td>260.74</td>
<td>479.97</td>
<td>683.86</td>
<td>1661.08</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time windows</th>
<th># loc.</th>
<th>length of time windows in min</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>0.59</td>
<td>0.64</td>
<td>0.56</td>
<td>0.91</td>
<td>0.72</td>
<td>3.42</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>1.70</td>
<td>0.97</td>
<td>1.90</td>
<td>1.25</td>
<td>1.67</td>
<td>7.49</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>2.62</td>
<td>2.40</td>
<td>5.99</td>
<td>1.67</td>
<td>5.54</td>
<td>18.22</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>1.76</td>
<td>2.18</td>
<td>1.55</td>
<td>1.69</td>
<td>1.37</td>
<td>8.55</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>0.72</td>
<td>0.86</td>
<td>1.20</td>
<td>1.00</td>
<td>2.43</td>
<td>6.21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>0.78</td>
<td>3.17</td>
<td>2.65</td>
<td>3.09</td>
<td>3.64</td>
<td>13.32</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>3.46</td>
<td>1.98</td>
<td>2.90</td>
<td>3.92</td>
<td>3.48</td>
<td>15.74</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>2.48</td>
<td>2.90</td>
<td>2.48</td>
<td>3.03</td>
<td>13.71</td>
<td>24.60</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>1.50</td>
<td>1.95</td>
<td>1.61</td>
<td>2.15</td>
<td>3.96</td>
<td>10.76</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>19.58</td>
<td>18.27</td>
<td>19.94</td>
<td>19.33</td>
<td>21.98</td>
<td>99.09</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>11.83</td>
<td>9.84</td>
<td>4.63</td>
<td>6.17</td>
<td>6.82</td>
<td>39.24</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>17.74</td>
<td>8.96</td>
<td>14.27</td>
<td>19.91</td>
<td>6.99</td>
<td>67.86</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>5.31</td>
<td>37.11</td>
<td>27.97</td>
<td>15.27</td>
<td>15.40</td>
<td>101.06</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>35.41</td>
<td>27.72</td>
<td>29.41</td>
<td>11.53</td>
<td>31.47</td>
<td>135.53</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>469.66</td>
<td>31.17</td>
<td>42.98</td>
<td>1509.76</td>
<td>112.51</td>
<td>2166.07</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>575.13</td>
<td>150.12</td>
<td>160.04</td>
<td>1600.60</td>
<td>231.27</td>
<td>2717.17</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th># loc.</th>
<th>length of time windows in min</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11</td>
<td>1.14</td>
<td>1.14</td>
<td>0.98</td>
<td>1.08</td>
<td>0.90</td>
<td>5.24</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1.53</td>
<td>2.73</td>
<td>2.03</td>
<td>1.67</td>
<td>14.23</td>
<td>22.18</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>2.76</td>
<td>2.03</td>
<td>2.70</td>
<td>2.23</td>
<td>9.66</td>
<td>19.38</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>1.90</td>
<td>3.32</td>
<td>1.76</td>
<td>2.15</td>
<td>1.97</td>
<td>11.11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1.73</td>
<td>1.81</td>
<td>1.62</td>
<td>1.59</td>
<td>1.67</td>
<td>8.42</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>2.12</td>
<td>4.48</td>
<td>3.01</td>
<td>2.23</td>
<td>9.47</td>
<td>21.31</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>3.59</td>
<td>3.43</td>
<td>6.36</td>
<td>4.48</td>
<td>4.24</td>
<td>22.11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>4.77</td>
<td>9.05</td>
<td>5.45</td>
<td>7.82</td>
<td>28.22</td>
<td>55.30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>2.18</td>
<td>2.54</td>
<td>2.57</td>
<td>2.95</td>
<td>6.12</td>
<td>16.36</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>26.05</td>
<td>49.33</td>
<td>27.96</td>
<td>23.15</td>
<td>21.92</td>
<td>148.40</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>34.68</td>
<td>10.90</td>
<td>12.31</td>
<td>7.30</td>
<td>17.74</td>
<td>82.93</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>20.59</td>
<td>14.95</td>
<td>19.00</td>
<td>19.56</td>
<td>8.77</td>
<td>82.87</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>43.87</td>
<td>21.75</td>
<td>58.72</td>
<td>17.85</td>
<td>48.23</td>
<td>190.41</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>56.69</td>
<td>52.90</td>
<td>33.45</td>
<td>19.59</td>
<td>71.15</td>
<td>233.78</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>71.43</td>
<td>50.95</td>
<td>43.91</td>
<td>1520.84</td>
<td>374.92</td>
<td>2062.05</td>
<td></td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>275.04</td>
<td>231.30</td>
<td>221.83</td>
<td>1634.48</td>
<td>619.20</td>
<td>2981.85</td>
<td></td>
</tr>
</tbody>
</table>

Table B.5.: Filter distance 500 km: Run times in seconds for the MILP model solution process
<table>
<thead>
<tr>
<th>base inst.</th>
<th># loc.</th>
<th>0</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>600</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>1.12</td>
<td>1.03</td>
<td>0.76</td>
<td>0.66</td>
<td>0.72</td>
<td>4.29</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>2.37</td>
<td>1.73</td>
<td>2.32</td>
<td>1.11</td>
<td>0.69</td>
<td>8.22</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>2.58</td>
<td>2.56</td>
<td>2.11</td>
<td>1.17</td>
<td>2.67</td>
<td>11.08</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.06</td>
<td>1.09</td>
<td>0.79</td>
<td>1.00</td>
<td>1.41</td>
<td>5.35</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>0.62</td>
<td>1.00</td>
<td>0.56</td>
<td>0.83</td>
<td>1.98</td>
<td>4.99</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>0.89</td>
<td>0.89</td>
<td>0.72</td>
<td>0.90</td>
<td>3.12</td>
<td>6.52</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>3.23</td>
<td>4.04</td>
<td>2.65</td>
<td>3.88</td>
<td>2.46</td>
<td>16.27</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>4.04</td>
<td>4.95</td>
<td>4.29</td>
<td>4.56</td>
<td>3.65</td>
<td>21.49</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>1.16</td>
<td>1.23</td>
<td>1.33</td>
<td>0.97</td>
<td>1.31</td>
<td>5.99</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
<td>17.68</td>
<td>14.37</td>
<td>16.76</td>
<td>10.31</td>
<td>21.17</td>
<td>80.28</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>5.69</td>
<td>9.16</td>
<td>3.46</td>
<td>4.18</td>
<td>5.05</td>
<td>27.55</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>5.72</td>
<td>7.85</td>
<td>6.05</td>
<td>5.76</td>
<td>16.61</td>
<td>42.00</td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>4.90</td>
<td>3.98</td>
<td>2.39</td>
<td>2.82</td>
<td>7.61</td>
<td>21.70</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>15.99</td>
<td>15.38</td>
<td>26.69</td>
<td>27.71</td>
<td>15.71</td>
<td>101.48</td>
</tr>
<tr>
<td>15</td>
<td>27</td>
<td>39.83</td>
<td>32.88</td>
<td>57.52</td>
<td>26.18</td>
<td>67.91</td>
<td>224.31</td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>106.88</td>
<td>102.13</td>
<td>128.41</td>
<td>92.03</td>
<td>152.07</td>
<td>581.51</td>
</tr>
<tr>
<td># time windows</td>
<td>2</td>
<td>11</td>
<td>0.59</td>
<td>0.62</td>
<td>0.58</td>
<td>0.89</td>
<td>0.70</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>0.84</td>
<td>0.87</td>
<td>0.97</td>
<td>0.98</td>
<td>1.36</td>
<td>5.02</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>2.54</td>
<td>1.33</td>
<td>3.49</td>
<td>1.19</td>
<td>3.68</td>
<td>12.23</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.37</td>
<td>1.54</td>
<td>1.11</td>
<td>1.56</td>
<td>1.58</td>
<td>7.16</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>0.70</td>
<td>0.89</td>
<td>1.17</td>
<td>1.00</td>
<td>2.44</td>
<td>6.19</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>1.08</td>
<td>2.61</td>
<td>2.36</td>
<td>1.55</td>
<td>3.04</td>
<td>10.62</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>3.63</td>
<td>3.84</td>
<td>5.41</td>
<td>3.73</td>
<td>1.95</td>
<td>18.57</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>2.45</td>
<td>2.92</td>
<td>2.47</td>
<td>3.01</td>
<td>13.73</td>
<td>24.57</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>1.59</td>
<td>1.01</td>
<td>1.28</td>
<td>2.71</td>
<td>7.97</td>
<td>7.97</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
<td>23.65</td>
<td>17.11</td>
<td>14.98</td>
<td>22.47</td>
<td>15.79</td>
<td>93.99</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>7.72</td>
<td>10.62</td>
<td>7.75</td>
<td>5.34</td>
<td>3.57</td>
<td>35.01</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>14.91</td>
<td>9.28</td>
<td>17.41</td>
<td>15.43</td>
<td>6.10</td>
<td>63.14</td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>3.85</td>
<td>3.62</td>
<td>7.21</td>
<td>5.37</td>
<td>13.29</td>
<td>33.34</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>35.44</td>
<td>27.77</td>
<td>29.42</td>
<td>11.45</td>
<td>30.61</td>
<td>134.69</td>
</tr>
<tr>
<td>15</td>
<td>27</td>
<td>40.20</td>
<td>49.20</td>
<td>31.31</td>
<td>141.82</td>
<td>73.94</td>
<td>336.48</td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>140.59</td>
<td>133.24</td>
<td>126.91</td>
<td>217.11</td>
<td>174.52</td>
<td>792.36</td>
</tr>
<tr>
<td># time windows</td>
<td>3</td>
<td>11</td>
<td>1.14</td>
<td>1.14</td>
<td>0.97</td>
<td>1.11</td>
<td>0.87</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>1.20</td>
<td>1.41</td>
<td>1.53</td>
<td>1.79</td>
<td>6.57</td>
<td>12.50</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>1.78</td>
<td>2.18</td>
<td>0.94</td>
<td>1.50</td>
<td>3.18</td>
<td>9.58</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1.50</td>
<td>2.40</td>
<td>1.27</td>
<td>1.78</td>
<td>1.76</td>
<td>8.70</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1.75</td>
<td>1.76</td>
<td>1.59</td>
<td>1.59</td>
<td>1.65</td>
<td>8.35</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>2.90</td>
<td>4.28</td>
<td>2.93</td>
<td>2.18</td>
<td>6.02</td>
<td>18.31</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>5.01</td>
<td>2.75</td>
<td>2.43</td>
<td>2.90</td>
<td>4.40</td>
<td>17.49</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>4.70</td>
<td>9.08</td>
<td>5.43</td>
<td>7.69</td>
<td>28.08</td>
<td>54.97</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>1.76</td>
<td>1.17</td>
<td>1.29</td>
<td>2.18</td>
<td>4.16</td>
<td>10.59</td>
</tr>
<tr>
<td>10</td>
<td>21</td>
<td>32.34</td>
<td>22.92</td>
<td>32.95</td>
<td>30.12</td>
<td>20.31</td>
<td>138.63</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>118.19</td>
<td>14.90</td>
<td>9.77</td>
<td>8.27</td>
<td>13.70</td>
<td>164.82</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>12.62</td>
<td>16.63</td>
<td>17.88</td>
<td>24.26</td>
<td>8.56</td>
<td>79.95</td>
</tr>
<tr>
<td>13</td>
<td>18</td>
<td>6.96</td>
<td>4.17</td>
<td>14.06</td>
<td>5.93</td>
<td>26.46</td>
<td>57.57</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>56.50</td>
<td>52.74</td>
<td>33.49</td>
<td>19.69</td>
<td>71.26</td>
<td>233.69</td>
</tr>
<tr>
<td>15</td>
<td>27</td>
<td>28.48</td>
<td>62.98</td>
<td>41.43</td>
<td>47.32</td>
<td>178.54</td>
<td>358.75</td>
</tr>
<tr>
<td>Σ</td>
<td></td>
<td>276.82</td>
<td>200.50</td>
<td>167.95</td>
<td>158.31</td>
<td>375.55</td>
<td>1179.12</td>
</tr>
</tbody>
</table>

Table B.6.: Filter distance 1000 km: Run times in seconds for the MILP model solution process
C. Heuristic Approaches - Pseudo code

The linear-time greedy algorithm of Lin et al. (2007) first computes \( LOW(i) \) and \( FAR(i) \) for every gas station \( i \in \{0,1,\ldots,n-2\} \) and then determines an optimal refueling plan (if such a plan exists) based on the corresponding function values. Given a series of gas stations \( \{0,1,\ldots,n-1\} \), the \( FAR \) function \( \text{FAR} : \{0,1,\ldots,n-2\} \rightarrow \{0,1,\ldots,n-1\} \) is defined specifying for each gas station \( i \) which gas station following \( i \) is the farthest reachable gas station with a fuel level equal to the maximum tank capacity.\(^87\) The \( LOW \) function \( \text{LOW} : \{0,1,\ldots,n-2\} \rightarrow \{0,1,\ldots,n-1\} \) determines for each gas station \( i \) the next gas station with a lower fuel price. For the formal definitions of the \( FAR \) and the \( LOW \) function we refer to Lin et al. (2007).

Although the algorithm of Lin et al. (2007) allows arbitrary start fuel levels, depending on the heuristic the algorithm is embedded in we recommend to add an artificial gas station for the start with a fuel consumption of \( \bar{T}^{\text{max}} - \bar{f}^{\text{start}} \) to reach the original start location and a fuel price of 0. If this option is chosen, the trip begins with a fuel level of 0 as described in Section 4.9 for the algorithm of Khuller et al. (2007). The modified gas station set \( \tilde{S}^{\text{stations}} = \{0,1,\ldots,n-1\} \) then contains the start and end locations of the original problem and the new artificial gas station. This is one possibility to allow an easy and early check whether the original problem is feasible. If the first real gas station \( i = 2 \) is not reachable with a full tank from the new artificial gas station \( \text{FAR}(0) < 2 \), the original problem is infeasible. This can be checked during the computation of the \( FAR \) function values.

The pseudo code for the determination of the \( FAR \) and \( LOW \) function values integrating the modification on the input data including the ones described in Section 4.9 is given by Algorithms 12 and 13. The two algorithms themselves are essentially adapted from Lin et al. (2007). Some footnotes give hints on the modifications to be made depending on the properties of the underlying heuristic which controls the choice of gas stations.

**Algorithm 12** Compute \( \text{FAR}(.) \)

**Input:**

\[
\begin{align*}
\bar{T}^{\text{max}} & \quad \text{Vehicle tank capacity minus reserve amount} \\
\tilde{S}^{\text{stations}} & \quad \text{Modified set of all gas stations} \ i \text{ along the route, } i = 0, \ldots, n - 1
\end{align*}
\]

\(^{87}\) Taking into account a reserve fuel quantity, the modified tank capacity \( \bar{T}^{\text{max}} \) is required to be considered here.
C. Heuristic Approaches - Pseudo code

(including artificial ones, i.e. including start and end locations and an additional gas station to map the start fuel level)

\[ \tilde{\Delta}_{\text{cons}}^{(i,i+1)} \]

Fuel consumption between two consecutive gas stations including consumptions for detours

The fuel consumption for the last arc is modified by adding \( \tilde{f}^{\text{end}} \), the consumption for the first arc is set to be equal to \( \tilde{T}^{\text{max}} - f^{\text{start}} \)

Output:

\[ \text{FAR}(i) \quad \forall \ i \in \tilde{S}_{\text{stations}} \setminus \{n - 1\} \text{ as array} \]

```
1: // Initialize
2: fuelInTank ← \( \tilde{T}^{\text{max}} \)
3: here ← 0
4:
5: for i = 0 to n - 2 do
6:     // Calculate FAR(i)
7:     while here < n - 1 \&\& fuelInTank > \( \tilde{\Delta}_{\text{cons}}^{(here,here+1)} \) do
8:         // Check the reachability of gas station here + 1
9:         fuelInTank ← fuelInTank - \( \tilde{\Delta}_{\text{cons}}^{(here,here+1)} \)
10:         here ← here + 1
11:     end while
12:     // here is the farthest reachable gas station from gas station i
13:     FAR(i) ← here
14:     // Determine the fuel level when reaching gas station here from gas station i
15:     if here > i then
16:         // At least the next gas station i + 1 is reachable from i consuming at most \( \tilde{T}^{\text{max}} \)
17:         // fuel.\(^{88}\)
18:         fuelInTank ← fuelInTank + \( \tilde{\Delta}_{\text{cons}}^{(i,i+1)} \)
19:     else
20:
21:     end if
22: end for
```

\(^{88}\) For \( i = 0 \) a check can be made here if \( i = 2 \) is reachable (in this case, \( here \geq 2 \)). If not, in the original problem the first gas station is not reachable with the start fuel level. Similar as in the case that \( here = i \) (else-case), the algorithm can either be stopped or continued.
// The next gas station $i + 1$ cannot be reached, that means the fuel consumption
// on the path between gas station $i$ and $i + 1$ exceeds $\tilde{T}_{max}$. In this case, there is
// no solution to the refueling subproblem.\textsuperscript{89}

$\text{fuelInTank} \leftarrow \tilde{T}_{max}$
$\text{here} \leftarrow i + 1$

end if

$i \leftarrow i + 1$

ddo

return the array of values $\text{FAR}(i), i \in \tilde{S}_{\text{stations}} \setminus \{n - 1\}$

---

**Algorithm 13** Compute $\text{LOW}(\cdot)$

**Input:**

$\tilde{S}_{\text{stations}}$ Modified set of all gas stations $i$ along the route, $i = 0, \ldots n - 1$
(including artificial ones, i.e. including start and end locations
and an additional gas station to map the start fuel level)

$\tilde{P}_i$ Fuel price at gas station $i \in \tilde{S}_{\text{stations}}$
The fuel price at gas station 0 (artificial vertex) and at
gas station $n - 1$ (original final destination, no gas station)
is set to be zero and at gas station 1 (original start location) to
infinity

**Output:**

$\text{LOW}(i) \quad \forall i \in \tilde{S}_{\text{stations}} \setminus \{n - 1\}$ as array

1: for $i = n - 2$ to 0 do
2:
3: // Calculate $\text{LOW}(i)$ for gas station $i$.
4:

\textsuperscript{89} One could either stop at this point and leave the heuristic, as no solution exists or go on to return detailed
information between which gas stations distances are too large including information on all gas stations. The
information could be returned to the heuristic to modify the gas station set in a sophisticated way. We decided
to keep the original version of Lin et al. (2007) and further proceed with the heuristic.
Algorithm 14 determines for every gas station $i \in \{0, 1, \ldots, n - 2\}$ the fuel consumption to the next gas station with lower fuel price. To determine the computational complexity, take into account that the test whether $next < LOW(i)$ in the while-loop is equivalent to the test if $\tilde{P}_{\text{next}} \geq \tilde{P}_i$ in the while-loop of Algorithm 13 and that there is only at least one iteration of the while-loop if $\tilde{P}_{i+1} \geq \tilde{P}_i$. Following the same arguments as Lin et al. (2007) for the proof of the linear complexity of the $LOW$ algorithm, Algorithm 14 has a complexity of $O(n)$, where $n$ is the number of gas stations.$^{90}$

Algorithm 14 Compute $\text{consLOW}(.)$

Input:

- $\tilde{S}_{\text{stations}}$ Modified set of all gas stations $i$ along the route, $i = 0, \ldots n - 1$ (including artificial ones, i.e. including start and end locations and an additional gas station to map the start fuel level)
- $\tilde{\Delta}_{\text{cons}(i,i+1)}$ Fuel consumption between two consecutive gas stations including

---

$^{90}$ Algorithms 13 and 14 could be merged to compute $LOW$ and $\text{consLOW}$ in one step (and for example return the results as an object) to improve performance. To highlight the differences from the original version, here, the computations are executed in two steps.
consumptions for detours
The fuel consumption for the last arc is modified by adding $\tilde{f}^{end}$, the consumption for the first arc is set to be equal to $\tilde{T}^{max} - \tilde{f}^{start}$

$LOW(i)$  The next gas station with lower fuel price for each $i \in \tilde{S}^{stations} \setminus \{n - 1\}$

Output:

$$consLOW(i) \quad \forall i \in \tilde{S}^{stations} \setminus \{n - 1\}$$

1: for $i = n - 2$ to 0 do
2:  // Calculate $consLOW(i)$ for gas station $i$.
3:  $consLOW(i) \leftarrow \Delta cons_{(i,i+1)}$
4:  $next \leftarrow i + 1$
5:  while $next < LOW(i)$ do
6:     $consLOW(i) \leftarrow consLOW(i) + consLOW(next)$
7:     $next \leftarrow LOW(next)$
8:  end while
9:  $i \leftarrow i - 1$
10: end for
11: return the array of values $consLOW(i), i \in \tilde{S}^{stations} \setminus \{n - 1\}$

With Algorithm 15 a modified version of the linear time algorithm of Lin et al. (2007) is presented to determine a refueling strategy respecting the restrictions described in Section 4.9 including the minimum purchase quantity. The first part of the while-loop is similar to the algorithm of Lin et al. (2007). The differences are the modifications made in case that the purchase amount determined in the first step is less than the minimum purchase quantity. Then, it is successively tried to make the modifications described in Section 4.9. The cases $LOW(i) > FAR(i)$ and $LOW(i) \leq FAR(i)$ are treated separately.
Algorithm 15 Determine refueling strategy

Input:

\( \tilde{S}^{\text{stations}} \) Modified set of all gas stations \( i \) along the route, \( i = 0, \ldots n - 1 \) (including artificial ones, i.e. including start and end locations and an additional gas station to map the start fuel level)

\( \tilde{T}^{\text{max}} \) Vehicle tank capacity minus reserve amount

\( \tilde{\Delta}_{\text{cons}}(i,i+1) \) Fuel consumption between two consecutive gas stations including consumptions for detours
The fuel consumption for the last arc is modified by adding the end fuel level, the consumption for the first arc is set to be equal to \( \tilde{T}^{\text{max}} - \tilde{f}^{\text{start}} \)

\( \tilde{P}_i \) Fuel price at gas station \( i \in \tilde{S}^{\text{stations}} \)
The fuel price at gas station 0 (artificial vertex) and at gas station \( n - 1 \) (original final destination, no gas station) is set to be zero and at gas station 1 (original start location) to infinity

Output:

Refueling policy as an array of refueling quantities at each gas station

\[ \Delta^{\text{refuel}} = (\Delta^{\text{refuel}}_0, \Delta^{\text{refuel}}_1, \ldots, \Delta^{\text{refuel}}_{n-2}) \]

1: Execute Algorithms 12, 13 and 14 to determine \( \text{LOW}(i) \), \( \text{FAR}(i) \) and \( \text{consLOW}(i) \)
2: for all \( i \in \tilde{S}^{\text{stations}} \setminus \{n - 1\} \)
3: 4: if \( \text{FAR}(0) < 2 \) then
5: 6: The first gas station in the original problem is not reachable. Memorize this information if relevant for the heuristic.\(^91\)
7: 8: 9: end if
10: 11: // Initialize
12: \( \Delta^{\text{refuel}} \leftarrow (0, 0, \ldots 0) \)
13: \( \text{fuelInTank} \leftarrow 0 \)

\(^91\) This if-statement is only necessary if the decision was made to process this information in the enclosing heuristic. If not, Algorithm 12 throws an error and the solution process for the refueling subproblem is left without solution.
\begin{verbatim}
14:  i ← 0
15:  helpRefuel ← 0
16:  consLOW(i) ← 0
17:  lastRefPos ← −1
18:  while i < n − 1 do
19:    if FAR(i) = i then
20:      Stop and report that the refueling subproblem is infeasible.\footnote{In that case, the gas station selection has to be modified. Information on which gas stations could not be left can be returned to the heuristic.}
21:    end if
22:    if LOW(i) > FAR(i) then
23:      helpRefuel ← \( \tilde{T}^{\max} − fuelInTank \)
24:      if helpRefuel ≥ \( \Delta^{\min} \) then
25:        \( \Delta_i^{\text{refuel}} \leftarrow \text{helpRefuel} \)
26:        lastRefPos ← i
27:      end if
28:      \( fuelInTank \leftarrow \tilde{T}^{\max} − \Delta_i^{\text{cons}} \)
29:      i ← i + 1
30:      continue
31:    else
32:      if consLOW(i) <= fuelInTank then
33:        \( fuelInTank \leftarrow fuelInTank − \text{consLOW}(i) \)
34:        i ← LOW(i)
35:        continue
36:      else
37:        helpRefuel ← \( \text{consLOW}(i) − fuelInTank \)
38:      end if
39:    end if
40:  end while
\end{verbatim}
if $\text{helpRefuel} \geq \Delta_{\text{min}}$ then

// Refuel the minimum amount to be able to reach gas station LOW(i).
$\Delta_{\text{refuel}} = \text{helpRefuel}$
$\text{lastRefPos} \leftarrow i$

// Move to gas station LOW(i) without refueling anymore.
$\text{fuelInTank} \leftarrow \text{fuelInTank} + \Delta_{\text{refuel}} - \text{consLOW}(i)$
$i \leftarrow \text{LOW}(i)$
continue

end if

end if

// This part of the algorithm is only reached if refueling should take place
at gas station $i$ but the refueling amount determined by the standard
// technique given by Lin et al. (2007) is less than the minimum purchase
// quantity.

if $\text{LOW}(i) > \text{FAR}(i)$ then

// With a refueling amount of $\text{helpRefuel}$ the tank capacity would be reached.

if $\Delta_{\text{cons}_{i,i+1}} > \text{fuelInTank}$ then

if lastRefPos > 0 then

modAmount $\leftarrow \Delta_{\text{min}} - \text{helpRefuel}$

if $\Delta_{\text{refuel}_{\text{lastRefPos}}} - \text{modAmount} \geq \Delta_{\text{min}}$ then

$\Delta_{\text{refuel}_{\text{lastRefPos}}} \leftarrow \Delta_{\text{refuel}_{\text{lastRefPos}}} - \text{modAmount}$
$\Delta_{\text{refuel}} \leftarrow \Delta_{\text{min}}$

else

$\Delta_{\text{refuel}} \leftarrow \Delta_{\text{refuel}_{\text{lastRefPos}}} + \text{helpRefuel}$
$\Delta_{\text{refuel}_{\text{lastRefPos}}} \leftarrow 0$

end if

$\text{fuelInTank} \leftarrow \text{fuelInTank} + \text{helpRefuel}$

end if

end if

end if
else

Stop and report that the refueling subproblem is infeasible if the minimum
purchase quantity has to be respected.\footnote{93}{Again, the heuristic can be informed.}
end if
end if

\(i \leftarrow i + 1\)

\(\text{fuelInTank} \leftarrow \text{fuelInTank} - \tilde{\Delta}_{(i,i+1)}^{\text{cons}}\)

else  \hspace{1em} // LOW(i) \leq \text{FAR}(i)

if \(\text{LOW}(i) = n - 1 \land \text{lastRefPos} > 0\) then

// Try to reach the final destination with a fuel level equal to the
// minimum end fuel level. This corresponds to a end fuel level of 0
// with the adjustments on the input data made.

modAmount \leftarrow \Delta_{\text{min}}^{\text{refuel}} - \text{helpRefuel}

if \(\Delta_{\text{lastRefPos}}^{\text{refuel}} - \text{modAmount} \geq \Delta_{\text{min}}^{\text{refuel}} \land \text{fuelInTank} - \text{modAmount} > 0\) then

\(\Delta_{\text{lastRefPos}}^{\text{refuel}} \leftarrow \Delta_{\text{lastRefPos}}^{\text{refuel}} - \text{modAmount}\)

\(\Delta_i^{\text{refuel}} \leftarrow \Delta_{\text{min}}\)

\(\text{fuelInTank} \leftarrow \text{fuelInTank} + \text{helpRefuel} - \text{consLOW}(i)\)

\(i \leftarrow n - 1\)

else if \(\text{fuelInTank} - \Delta_{\text{lastRefPos}}^{\text{refuel}} > 0\) then

\(\Delta_i^{\text{refuel}} \leftarrow \Delta_{\text{lastRefPos}}^{\text{refuel}} + \text{helpRefuel}\)

\(\Delta_{\text{lastRefPos}}^{\text{refuel}} \leftarrow 0\)

\(\text{fuelInTank} \leftarrow \text{fuelInTank} + \text{helpRefuel} - \text{consLOW}(i)\)

\(i \leftarrow n - 1\)

end if

else if \(\text{fuelInTank} + \Delta_{\text{min}}^{\text{min}} \leq \tilde{T}^{\text{max}}\) then

\(\text{fuelInTank} \leftarrow \text{fuelInTank} + \Delta_{\text{min}}^{\text{min}} - \text{consLOW}(i)\)

\(i \leftarrow \text{LOW}(i)\)
We assume that the modified vehicle tank capacity $\tilde{T}^{\text{max}}$ is at least three times as large as the minimum purchase quantity. If this is the case the modifications made in an iteration for the last gas station prior to $i$ with positive refueling still lead to a feasible solution. Otherwise, the fuel level determined for the arrival at the gas station currently considered may be negative.
Proof:
To show that these modifications are always feasible if the modified vehicle tank capacity \( \tilde{T}_{\text{max}} \) is at least three times as large as the minimum purchase quantity, at first consider the case that the refueling amount at the previous gas station with positive refueling amount is modified by the amount \( \text{modAmount} \). Note that \( \text{modAmount} = \Delta_{\text{min}} - \text{helpRefuel} \) holds and in all cases where this modification is made and the last gas station \( n - 1 \) is not reachable, \( \text{fuelInTank} + \Delta_{\text{min}} > \tilde{T}_{\text{max}} \) is true.\(^{94}\) In case that the modification at the previous gas station with positive refueling amount would lead to a negative fuel level for the arrival at the current gas station considered, the modified tank capacity has to be less than two times the minimum purchase quantity (\( \tilde{T}_{\text{max}} < 2 \Delta_{\text{min}} \)):

\[
\text{fuelInTank} - \text{modAmount} < 0 \\
\Leftrightarrow \text{fuelInTank} < \text{modAmount} \\
\Rightarrow \tilde{T}_{\text{max}} - \Delta_{\text{min}} < \text{modAmount} \quad \text{as} \quad \text{fuelInTank} + \Delta_{\text{min}} > \tilde{T}_{\text{max}} \\
\Rightarrow \tilde{T}_{\text{max}} - \Delta_{\text{min}} < \Delta_{\text{min}} - \text{helpRefuel} \quad \text{as} \quad \text{modAmount} = \Delta_{\text{min}} - \text{helpRefuel} \\
\Rightarrow \tilde{T}_{\text{max}} - \Delta_{\text{min}} < \Delta_{\text{min}} \quad \text{as} \quad \text{helpRefuel} > 0 \\
\Leftrightarrow \tilde{T}_{\text{max}} < 2 \Delta_{\text{min}}
\]

If the previous gas station with positive refueling amount is removed by the algorithm, in case that \( \text{LOW}(i) < n - 1 \) we know that \( \text{fuelInTank} + \Delta_{\text{min}} > \tilde{T}_{\text{max}} \) and \( \Delta_{\text{lastRefPos}} - \text{modAmount} < \Delta_{\text{min}} \) as otherwise, this decision is not taken. The elimination of the refueling stop at gas station \( \text{lastRefPos} \) leads to a negative amount of fuel in the tank at the current gas station considered if \( \text{fuelInTank} - \Delta_{\text{lastRefPos}} < 0 \). We can conclude that in this case \( \tilde{T}_{\text{max}} < 3 \Delta_{\text{min}} \) holds:

\[
\text{fuelInTank} - \Delta_{\text{lastRefPos}} < 0 \\
\Leftrightarrow \text{fuelInTank} < \Delta_{\text{lastRefPos}} \\
\Rightarrow \tilde{T}_{\text{max}} - \Delta_{\text{min}} < \Delta_{\text{lastRefPos}} \quad \text{as} \quad \text{fuelInTank} + \Delta_{\text{min}} > \tilde{T}_{\text{max}} \\
\Rightarrow \tilde{T}_{\text{max}} - \Delta_{\text{min}} < \Delta_{\text{min}} + \text{modAmount} \quad \text{as} \quad \Delta_{\text{lastRefPos}} - \text{modAmount} < \Delta_{\text{min}} \\
\Rightarrow \tilde{T}_{\text{max}} - \Delta_{\text{min}} < 2 \Delta_{\text{min}} \quad \text{as} \quad \text{modAmount} = \Delta_{\text{min}} - \text{helpRefuel} < \Delta_{\text{min}} \\
\Leftrightarrow \tilde{T}_{\text{max}} < 3 \Delta_{\text{min}}
\]

Without loss of generality, we assume that the start and the minimum end fuel level respect

\(^{94}\) In case that the final gas station \( n - 1 \) is reachable, a check is made whether the modification would lead to a negative fuel level when arriving at the current gas station considered.
C. Heuristic Approaches - Pseudo code

the reserve quantity.

The pseudo code of the construction heuristic described in Section 4.9 that can be applied to find a start solution is presented with Algorithm 16. To consider a minimum purchase quantity, a similar modification method can be used as in Algorithm 15. Note that this modification may especially be of interest if a very large end fuel level is chosen. In this case, the last gas station preceding the final destination may be reached with a relatively high fuel level but still, refueling is necessary to reach the final destination with at least the required amount. With a relatively high density of gas stations considered for refueling, other cases which may trigger the modification may not occur. Thus, in general, the construction heuristic should find a solution if one exists.

**Algorithm 16** Construct initial refueling plan

**Input:**

\[
\tilde{S}_{\text{stations}} \quad \text{Modified set of all gas stations } i \text{ along the route, } i = 0, \ldots, n - 1 \\
\quad \text{(including artificial ones, i.e. including start and end locations} \\
\quad \text{and an additional gas station to map the start fuel level)} \\
\tilde{T}_{\text{max}} \quad \text{Vehicle tank capacity minus reserve amount} \\
\tilde{\Delta}_{\text{cons}} \quad \text{Fuel consumption between two consecutive locations} \\
\quad \text{(without detours).} \\
\quad \text{The fuel consumption for the last arc } (n - 2, n - 1) \text{ is modified by} \\
\quad \text{adding the end fuel level (minus the reserve amount)} \\
\tilde{\Delta}_{0,n-1} \quad \text{Fuel consumption between start and end locations (without detours)} \\
\quad \text{with the modification for the end fuel level described above} \\
\tilde{\Delta}_{\text{consTo}} \quad \text{Fuel consumption for the detour from the route to gas station } i \\
\tilde{\Delta}_{\text{consFrom}} \quad \text{Fuel consumption for the detour from gas station } i \text{ back to the} \\
\quad \text{route}
\]

**Output:**

Feasible refueling plan as array of refueling quantities at each gas station

\[
\Delta_{\text{refuel}} = (\Delta_{0\text{refuel}}, \Delta_{1\text{refuel}}, \ldots, \Delta_{n-2\text{refuel}})
\]

1: // Initialize 
2: \(\Delta_{\text{0refuel}} = \tilde{T}_{\text{max}}\) 
3: \(\text{fuelInTank} \leftarrow \tilde{T}_{\text{max}}\) 
4: \(\text{minConsRemaining} \leftarrow \tilde{\Delta}_{\text{cons}}\)
5: $i \leftarrow 0$
6:
7: while $i < n - 1$ do
8: 
9:   $\text{maxReachableStation} \leftarrow i$
10:   $\text{routeConsRemaining} \leftarrow \text{minConsRemaining}$
11:   $\text{testFuelLevel} \leftarrow \text{fuelInTank}$
12:  
13:   for $j = i$ to $j < n - 1$ do
14:     
15:     $\text{testFuelLevel} \leftarrow \text{testFuelLevel} - \bar{\Delta}_{i,i+1} \text{cons}$
16:     $\text{routeConsRemaining} \leftarrow \text{routeConsRemaining} - \bar{\Delta}_{i,i+1} \text{cons}$
17:   
18:   if $\text{testFuelLevel} < 0$ then
19:     
20:     break
21:   
22:   end if
23:  
24:   if $j = n - 2$ then
25:     
26:     $\text{maxReachableStation} \leftarrow j + 1$
27:     $\text{fuelInTank} \leftarrow \text{testFuelLevel}$
28:   
29:   else if $\left( \text{testFuelLevel} - \bar{\Delta}_i \text{consTo} \geq 0 \\
30:     \wedge \text{routeConsRemaining} + \bar{\Delta}_i \text{consFrom} < \text{minConsRemaining} \right)$ then
31:     
32:     $\text{maxReachableStation} \leftarrow j + 1$
33:     $\text{minConsRemaining} \leftarrow \text{routeConsRemaining} + \bar{\Delta}_i \text{consFrom}$
34:     $\text{fuelInTank} \leftarrow \text{testFuelLevel}$
35:   
36:   end if
37:  
38:   $j \leftarrow j + 1$
39:  
40: end for
41:  
42: if $\text{maxReachableStation} = i$ then
43:     
44:     Stop and report that no solution was found.
45:  
46: end if
47:  
48: $i \leftarrow \text{maxReachableStation}$
49:  
50: if $i < n - 1$ then
C. Heuristic Approaches - Pseudo code

```
50:    fuelInTank ← fuelInTank − \( \bar{\Delta}_{consTo} \)
51:    helpRefuel ← \( T_{max} \) − fuelInTank
52:    if helpRefuel < \( \bar{\Delta}_{min} \) then
53:        modAmount ← \( \bar{\Delta}_{min} \) − helpRefuel
54:        if lastRefPos > 0 ∧ \( \Delta^{refuel}_{lastRefPos} \) − modAmount ≥ \( \bar{\Delta}_{min} \) then
55:            \( \Delta^{refuel}_{lastRefPos} \) ← \( \Delta^{refuel}_{lastRefPos} \) − modAmount
56:            \( \Delta^{refuel}_{i} \) ← \( \bar{\Delta}_{min} \)
57:            fuelInTank ← fuelInTank + helpRefuel
58:        else
59:            Stop and report that no solution was found.
60:    end if
61:    if lastRefPos > 0 ∧ \( \Delta^{refuel}_{lastRefPos} \) − modAmount ≥ \( \bar{\Delta}_{min} \) then
62:        \( \Delta^{refuel}_{lastRefPos} \) ← \( \Delta^{refuel}_{lastRefPos} \) − modAmount
63:        \( \Delta^{refuel}_{i} \) ← \( \bar{\Delta}_{min} \)
64:        fuelInTank ← fuelInTank + helpRefuel
65:    end if
66:    fuelInTank ← fuelInTank − \( \bar{\Delta}_{consFrom} \)
67:    lastRefPos ← i
68: end if
69: end while
70: return the refueling array \( \Delta^{refuel} = (\Delta^{refuel}_{0}, \Delta^{refuel}_{1}, \ldots, \Delta^{refuel}_{n-2}) \)
```
Bibliography


Bousonville, T., Dirichs, M., Hartmann, A., and Melo, T. (2012). Integrating information from heterogeneous data sources to improve decision making in the long-haul freight business. 6th International Symposium "Networks for Mobility".


