Optimal Integration of Autonomous Vehicles in Car Sharing

Development of a Heuristic considering Multimodal Transport and Integration in an Optimal Framework

Martin Sperr

31. März 2017

Abstract

In this thesis, we examine the potential impact of autonomous vehicles on car sharing. We examine the required fleet size and the arising cost in a deterministic model, if customers use multimodal transport. We create a mixed-integer linear program that extends the Vehicle Scheduling Problem considering fuel constraints and multimodal transport. We decompose the problem into smaller instances and solve them successively to gain a heuristical solution. For this, we examine two splitting approaches. Further, we create an improvement heuristic. Additionally, we apply Dantzig-Wolfe decomposition and solve the problem to optimality by a Branch-and-Price procedure. We investigate methods for solving the subproblems efficiently and create problem-dependent branching rules. This thesis concludes with an evaluation of the presented methods.

Zusammenfassung

In dieser Masterarbeit beschäftigen wir uns mit dem Potential, das Autonomes Fahren für Car Sharing hat. Wir untersuchen die erforderliche Anzahl an Fahrzeugen und die auftretenden Kosten, wenn die Kunden multimodalen Transport benutzen. Aufbauend auf dem Vehicle Scheduling Problem erstellen wir ein gemischt-ganzzahliges lineares Optimierungsproblem, das die begrenzte Reichweite und den multimodalen Transport berücksichtigt. Wir zerlegen das Problem in kleinere Instanzen und lösen diese hintereinander, um eine heuristische Lösung zu erreichen. Dafür untersuchen wir zwei Zerlegungsansätze. Zusätzlich erstellen wir eine Heuristik, die eine vorhandene Lösung verbessert. Außerdem wenden wir eine Dantzig-Wolfe-Zerlegung an und lösen das Problem optimal mit einem Branch-and-Price-Verfahren. Wir analysieren Methoden um die Teilprobleme effizient zu lösen und erstellen Branching-Regeln, die auf das Problem aufbauen. Den Abschluss der Arbeit bildet eine Auswertung der entwickelten Methoden.

Contents

Chapter 1

Introduction

FiXme Note: Check linebreak for in-text formulas

In this thesis, we extend an already developed heuristic for the routing of autonomous vehicles. The thesis is based on two master's theses [Kai16] and [Kno16] which were created earlier at the same department. The routing can be used for introducing autonomous vehicles into car sharing.

In commercial car sharing, a customer rents a car for a limited period of time. In the classic version, the customer gets the car on a fixed location and returns it to this location after usage. In contrast to this, free-floating car sharing allows the customer to pick up any car where this is available and to park it somewhere in the operation area. The customer usually books the car beforehand, typically via a smartphone application. He pays a certain amount per minute of car usage. This method is obviously more customer-friendly since the customer has no effort in getting to and from the renting location. But this means significantly more effort for the car sharing supplier. He has to provide a comprehensive offer of available cars, such that there is always a car where the customer needs it. Further he is responsible for refueling and servicing the cars, wherever they are. Customers may park their car where it suits them and simultaneously only rent a car if it is within a small walking distance to their current position. Therefore, the distribution of the cars heavily depends on the customer behavior. This might lead to an imbalance of supply and demand.

A possible solution for this is the usage of autonomous vehicles. Although they are not available on present day, this topic is highly researched. Autonomous vehicles may be available within the next ten to twenty years (cf. [Hau15]). The obvious advantage of autonomous cars is that they do not need a driver. An autonomous car is able to change the position after satisfying a customer on its own. The car can drive to a refuel station or to a position where it is needed next. For the customer, this behavior is similar to calling a taxi. The car picks him up on his present location and takes him to his destination. The supplier profits since he does not need employees for refueling or relocating the cars.

Besides cars, it is often advantageous for the customer to use public transport. For suitable trips, the usage of public transport is often faster and particularly cheaper. It is further more efficient in a city with many cars. On the other hand, the usage of public transport yields some inconveniences for the customer. The next station may be

too far away for walking or unfavorable changing times increase the total travel time. In these cases it is often a good idea to combine car sharing and public transport in one journey. The customer uses the car for driving to a station with a good connection and continues with using public transport. Therefore, the complete journey is divided into parts which we call „leg“. In each leg, the customer drives a certain distance without changing means of transport. The combination of different types of transport in one journey is called „multi-leg“.
The introduction of autonomous cars bears great potential for improvement for the car sharing supply. It involves huge changes for the maintenance of the vehicle fleet. The fleet means the number of vehicles that a car sharing supplier provides in a certain operation area. In order to estimate the profitability of introducing autonomous cars, the supplier is highly interested in the size of a vehicle fleet that is sufficient to maintain the car sharing supply. Therefore, we aim to find an optimal fleet size for the car sharing provider in this thesis. For this a small number of vehicles and a small driven distance in total is aspired, while still a good service shall be provided to the customer. Since the introduction of autonomous vehicles involves a great alteration in the service provided to the customer, it is hard to predict the change of the customer behavior. Therefore, we use current renting data in order to model the customer demand. With this we can estimate the improvement of autonomous vehicles compared to the current situation.
As mentioned at the beginning, the problem setting of this thesis is based on two previous theses. [Kai16] and [Kno16] examine simplified versions of autonomous vehicle routing. The routes are restricted to have only a single leg there. While [Kno16] provide fast heuristical solution methods via a time-dependent splitting of the trip set, [Kai16] develop an approach to solve this problem to optimality via a branch-and-price process. In this thesis, the heuristical methods are extended in order to cope with multi-leg routes. The goal is the determination of a good initial solution. With adapting the optimal approach and using the computed initial solution, an optimal solution for autonomous vehicle routing is determined.

Chapter 2

Problem Description and Classification

In this chapter, the problem is stated in detail and the notation is introduced. The problem is classified by relating it to known problems in literature and its complexity is determined. Finally, known approaches to similar problems are regarded. Most of the following considerations are already part of the underlying theses [Kai16] and [Kno16], except for the fact that multiple legs are allowed. All crucial results are repeated here for clarity.

2.1 Situation and Issue

We regard the situation of free-floating car sharing as it exists today in combination with autonomous vehicles. Free-floating car sharing means that a customer can rent an available car wherever and whenever one is available and use it as long as he needs. After usage, he parks the car somewhere in the operation area. We assume the existence of autonomous vehicles which behave the same as if a human were driving, but without a human being necessarily present. Instead of looking for a car, a customer books a car via a smartphone application and gets picked up by the car at the desired start location at the desired start time. For the customer, this would be similar to a taxi service.
The car sharing issue is combined with public transport as it is known today. There is a fixed schedule, according to which the bus or train visits public transport stations in a row at certain time points. A possible route for a customer may look as follows: The customer is picked up at his start position by a car and is brought to a station where he gets on a train. After finishing the train trip, he is picked up again by another car and is brought to his destination. It is also possible to change trains during this public transport trip. This behavior is very advantageous for the customer. While a partial train trip is cheaper than a pure car trip, the combination of cars and trains is faster than pure public transport since it does not require walking and transfer time.

Assumption of Perfect Information

As mentioned in Chapter 1, the introduction of autonomous vehicles probably involves a huge change in the customer behavior. The estimation of this is not part of the thesis, the focus lies rather in the potential of autonomous cars. Therefore, we use renting data from present time in order to model the customer behavior. The determined results for the optimal fleet size with autonomous vehicles can then be used to compare it to the current fleet size with conventional vehicles in order to estimate the possible enhancements. Further, we do not try to model an online behavior for the routing where the customer requests arise during during runtime. We rather assume to have perfect information of the customer behavior. This means we know all the travel requests in advance and create a vehicle schedule for the complete instance.

2.2 Problem Description and Notation

We aim to state the problem for the routing of autonomous vehicles such that it suits to the situation as described before. In order to realize this, we introduce a formal notation. We are given a set of customers where we know the travel requests for each of them. Each of these travel requests can be realized by one of a set of alternative multimodal routes that is also given in advance. Each multimodal route consists of a sequence of trips. In this context, a trip is either a car trip or a public transport trip and has a fixed start and end position as well as a fixed start and end time and is completed without interruption and with the same means of transport for the whole duration. Fulfilling a route means that the customer takes all the trips of this route in a row, this means he starts at the start point of the first trip and is finished at the end point of the last trip. The transition between two subsequent trips is the changing from a car to a train or the other way round. Each customer has to be satisfied, this means it is possible that he fulfills one of his alternative routes. We call the constraints ensuring the customer satisfaction „cover constraints“.
Our goal is to create a schedule for the vehicles of the car sharing supplier. For this we assume that there is already a schedule for public transport fixed. The car trips are created in such a way that they suit to the public transport schedule which we describe in Chapter 6. Therefore we assume the given routes to be feasible. Since we are interested in a schedule for the cars and the car trips are chosen appropriately, the public transport trips are not part of the input.
For fulfilling the trips, we have a set of vehicles. For each vehicle, we have a position where it starts and a time from when it is available. A vehicle can drive from its start point to a trip's start point, executing this trip, and then drive from the trip's end point to the next trip's start point. After fulfilling its last trip, the car stays at the end point of the last trip. The sequence, in which the vehicle executes the trips, is

called the duty of the vehicle.
A further restriction to the problem is that the vehicles have a maximal range. They can drive only a certain distance without refueling. We are given a set of refuel points where each car is able to refuel between serving trips. We call the constraints ensuring feasible fuel states for the vehicles „fuel constraints“.

Choice of the Multimodal Routes

As mentioned before, each customer has a set of multimodal routes one of which has to be fulfilled. In real life could be assumed that the customer chooses his alternative on its own, for example according to time reasons, costs or his personal preferences. In contrast, we assume in this context that the choice of the route is made by the system. This means each customer takes exactly this route which is necessary for the overall schedule to be optimal. This behavior can be interpreted in two ways: Either this problem setting aims to find the best possible schedule under assumption of ideal customer decisions or the car sharing supplier presents only one alternative to the customer for his travel request.
As described later more formally, we introduce costs for the route choice into the model. With these route costs we can model the customer preferences. The customer preferences contain for example the total travel time, the number of changes or the cost for the customer. They work as penalty costs for inconvenient route choices. This means a route that is disadvantageous for the customer is penalized. Then either the for the customer less favorable route is chosen if it fits better into the schedule and is penalized. Or the more favorable route is chosen although it suits not so good in the schedule.
We can further add the costs for public transport to the route costs. Although we regard the problem solely from the supplier's point of view, we motivate this approach as follows: If we assume that each customer chooses a pure public transport route, this schedule would be optimal since there arise no costs for the supplier. Obviously, this is not optimal in reality since the supplier achieves no profit. If we try to maximize the profit, the optimal schedule would contain long car distances, although a customer would not pay for this in reality. Thus we include the customer costs in the model.
In summary, we try to model additional customer costs and customer inconveniences as penalty terms in order to receive a more realistic customer behavior.

Customers, Trips and Vehicles

We first define the car trips, the multimodal routes, the customers and the connections between them.

Definition 1 (Trips, routes and customers)**.**

1. We are given a set of car trips $\mathcal{T}$. Each trip $t \in \mathcal{T}$ has a start and end location $p_t^{\text{start}}, p_t^{\text{end}}$ and a start and end time $z_t^{\text{start}}, z_t^{\text{end}}$.
2. Further, we are given a set of multimodal routes $\mathcal{M}$. A route $m = (t_1, \dots, t_{k_m})$ is a finite sequence of trips with the following properties:
$$p_{t_i}^{\text{end}} = p_{t_{i+1}}^{\text{start}} \qquad z_{t_i}^{\text{end}} \leq z_{t_{i+1}}^{\text{start}} \qquad \text{for all } i \in [k_m - 1]$$
We define the route start and end locations and times for $m \in \mathcal{M}$ as
$$p_m^{\text{start}} := p_{t_1}^{\text{start}} \qquad p_m^{\text{end}} := p_{t_{k_m}}^{\text{end}} \qquad z_m^{\text{start}} := z_{t_1}^{\text{start}} \qquad z_m^{\text{end}} := z_{t_{k_m}}^{\text{end}}.$$
3. We are given a set of customers $\mathcal{C}$. Each customer $c \in \mathcal{C}$ has a finite set of alternative multimodal routes.
4. Each trips belongs to exactly one route and each route belongs to exactly one customer. The mapping $M : \mathcal{T} \to \mathcal{M}$ indicates to which route a trip belongs and the mapping $C : \mathcal{M} \to \mathcal{C}$ shows to which customer a route belongs.
5. For each route of the same customer $m \in C^{-1}(c)$, the start and end positions are the same, but the start and end times may differ. We define the customer start and end times for $c \in \mathcal{C}$
$$z_c^{\text{start}} := \min_{t \in (C \circ M)^{-1}(c)} z_t^{\text{start}} \qquad z_c^{\text{end}} := \max_{t \in (C \circ M)^{-1}(c)} z_t^{\text{start}}.$$

We use the following notation in order to describe the preimages of the mappings M and C

$$\begin{aligned} C^{-1}(c) &:= C^{-1}(\{c\}) = \{m \in \mathcal{M} \mid C(m) = c\} && \text{for } c \in \mathcal{C} \\ M^{-1}(m) &:= M^{-1}(\{m\}) = \{t \in \mathcal{T} \mid M(t) = m\} && \text{for } m \in \mathcal{M} \\ (M \circ C)^{-1}(c) &:= M^{-1}\left(C^{-1}(c)\right) = \{t \in \mathcal{T} \mid C(M(t)) = c\} && \text{for } c \in \mathcal{C} \end{aligned}$$

for all routes of a customer, all trips of a route and all trips of a customer, respectively. The vehicles that are needed for fulfilling the trips are introduced as follows:

Definition 2 (Vehicles)**.** We are given a set of vehicles $\mathcal{V}$. For each vehicle $v \in \mathcal{V}$ we are given a start position p_v and a start time z_v.

Fuel and Refueling

We have to consider fuel restrictions. Fuel can be any form of energy with which the considered vehicle is powered. For each vehicle, the fuel level is in the interval $[0,1]$, where 1 means full capacity and 0 is empty. We call a drive without a customer, i. e. a drive between two trips, a deadhead trip. A car may visit a refuel station only during a deadhead trip. For simplicity of the model, each car is allowed to refuel at most once between two trips. On a refuel station, there are no capacity constraints, i. e. two or more vehicles may refuel at the same time at the same station. We define the refuel points and the fuel consumption as follows:

Definition 3 (Fuel and refuel points)**.**

1. We are given a set of refuel stations $\mathcal{R}$. Each refuel station $r \in \mathcal{R}$ has a location p_r.
2. We define $f^{\mathrm{d}}_{s,t}$ for $s \in \mathcal{V} \cup \mathcal{T} \cup \mathcal{R}, t \in \mathcal{T} \cup \mathcal{R}$ as the amount the fuel level decreases along the deadhead trip between s and t. We define f^{t}_t for $t \in \mathcal{T}$ as the amount of fuel a vehicle needs for a trip and f^0_v for $v \in \mathcal{V}$ as the initial fuel state of a vehicle.

The amount of fuel that is charged at a refuel point between two trips can be determined from the time the vehicle stays at the refuel point between these trips.

FiXme Note: ft for r in chapter 3

Ordering of the Trips

We define the time a vehicle needs to get from position p_1 to p_2 as t_{p_1,p_2}. We define

$$t_{s,t} = \begin{cases} t_{p_s^{\mathrm{end}}, p_t^{\mathrm{start}}} & \text{if } s,t \in \mathcal{T} \\ t_{p_s, p_t^{\mathrm{start}}} & \text{if } s \in \mathcal{V} \cup \mathcal{R}, t \in \mathcal{T} \\ t_{p_s^{\mathrm{end}}, p_t} & \text{if } s \in \mathcal{T}, t \in \mathcal{R} \\ t_{p_s, p_t} & \text{if } s \in \mathcal{V}, t \in \mathcal{R} \end{cases}$$

as the time a vehicle needs from one trip to another.
In order to decide whether a vehicle is able to fulfill two trips in a row, we define a partial ordering on the set of vehicles and trips. The set of public transport trips is left out in this definition.

Definition 4 (Order of trips)**.** The binary relation $\prec$ on $\mathcal{V} \cup \mathcal{T}$ is defined as follows:

$$\begin{aligned} s \prec t \quad &:\Leftrightarrow \quad z_s + t_{s,t} \le z_t^{\mathrm{start}} && \text{for all } s \in \mathcal{V}, t \in \mathcal{T} \\ s \prec t \quad &:\Leftrightarrow \quad \left(z_s^{\mathrm{end}} + t_{s,t} \le z_t^{\mathrm{start}}\right) \wedge ((M \circ C)(s) \neq (M \circ C)(t) \vee M(s) = M(t)) \\ & && \text{for all } s \in \mathcal{T}, t \in \mathcal{T} \\ s \not\prec t & && \text{for all } s \in \mathcal{V} \cup \mathcal{T}, t \in \mathcal{V} \end{aligned}$$

The binary relation $\preceq$ on $\mathcal{V} \uplus \mathcal{T}$ is defined as:

$$s \preceq t \quad :\Leftrightarrow \quad s = t \vee s \prec t \qquad\qquad \text{for all } s, t \in \mathcal{V} \uplus \mathcal{T}$$

The expression $s \prec t$ means that one car is able to fulfill both trips, first s and then t. A car must not cover two trips of the same customer, except when they belong to the same route. This results from the problem description, where for each customer exactly one route is fulfilled.

Remark 1. Note that $\preceq$ is not a partial order on $\mathcal{V} \uplus \mathcal{T}$ since the transitivity is missing. Let $t_1, t_2, t_3 \in \mathcal{T}$ with

$$z_{t_1}^{\text{end}} + t_{t_1,t_2} \leq z_{t_2}^{\text{start}} \qquad\qquad z_{t_2}^{\text{end}} + t_{t_2,t_3} \leq z_{t_3}^{\text{start}}$$

and

$$(M \circ C)(t_1) = (M \circ C)(t_3) \qquad (M \circ C)(t_1) \neq (M \circ C)(t_2) \qquad M(t_1) \neq M(t_3)\,.$$

Then follows $t_1 \preceq t_2 \preceq t_3$ and $t_1 \npreceq t_3$.

Problem Description and Cost

Using the previously introduced notation, we can formally define the problem of routing autonomous vehicles with fuel constraints and multi-leg cover constraints:

Definition 5 (Feasible schedule)**.** A feasible schedule is an assignment of vehicles to duties. Each vehicle $v \in \mathcal{V}$ is assigned to a duty $d(v)$. Each duty has the form

$$d(v) = ((r_1, t_1), \ldots, (r_{k_v}, t_{k_v}))$$

with $t_i \in \mathcal{T}$ and $r_i \in \mathcal{R} \uplus \{\emptyset\}$. An empty duty, i. e. $k_v = 0$, is feasible. Each customer $c \in \mathcal{C}$ is assigned to a route $m(c) \in \mathcal{M}$ with $C(m(c)) = c$. The schedule fulfills the following properties:

1. Time Feasibility: Let $v \in \mathcal{V}$ be arbitrary with duty $d(v)$. Using $t_0 := v$ holds for all $i \in [k_v]$:

$$t_{i-1} \prec t_i \qquad\qquad \text{if } r_i \neq \emptyset: \qquad\qquad z_{t_{i-1}}^{\text{end}} + t_{t_{i-1},r_i} + t_{r_i,t_i} \leq z_{t_i}^{\text{start}}$$

2. Fuel Feasibility: The fuel state of a vehicle considering $f^0, f^{\text{t}}, f^{\text{d}}$ and the respective refueling is always in the range $[0, 1]$.

3. Cover Constraints: For each customer $c \in \mathcal{C}$ holds:

$$m(c) \subseteq \bigcup_{v \in \mathcal{V}} d(v)$$

We write $r_i = \emptyset$ if no refuel point is visited between t_{i-1} and t_i.
We define the following types of costs for a feasible schedule according to Definition 5:

1. Vehicle cost c^{v}: unit cost for each vehicle used
2. Deadhead cost $c^{\mathrm{d}}_{s,t}$ for $s \in \mathcal{V} \cup \mathcal{T} \cup \mathcal{R}, t \in \mathcal{T} \cup \mathcal{R}$:
   cost if a vehicle drives to a trip or a refuel station without a customer using it
3. Trip cost c^{t}_t for $t \in \mathcal{T}$: cost for fulfilling a trip
4. Route cost c^{r}_m for $m \in \mathcal{M}$: cost for fulfilling a route

The vehicle costs are unit costs for each vehicle that is used to fulfill trips. If a vehicle does not serve any trips, i. e. its duty is empty, then no vehicle costs occur for this vehicle. The deadhead costs arise for all the deadhead trips that some vehicle takes. The trip costs and route costs arise if this trip or route is fulfilled.

Definition 6 (Schedule cost)**.** Let S be a schedule according to Definition 5 with $v \mapsto d(v)$ and $c \mapsto m(c)$.

1. For each non-empty duty, i. e. $d(v) \neq \emptyset$, we define the duty cost as

$$\operatorname{cost}(d(v)) := c^{\mathrm{v}} + \sum_{\substack{i\in[k_v]\\ r_i=\emptyset}} \left(c^{\mathrm{d}}_{t_{i-1},t_i} + c^{\mathrm{t}}_{t_i}\right) + \sum_{\substack{i\in[k_v]\\ r_i\neq\emptyset}} \left(c^{\mathrm{d}}_{t_{i-1},r_i} + c^{\mathrm{d}}_{r_i,t_i} + c^{\mathrm{t}}_{t_i}\right).$$

2. We define the schedule cost and the number of duties as

$$\operatorname{cost}(S) := \sum_{\substack{v\in\mathcal{V}\\ d(v)\neq\emptyset}} \operatorname{cost}(d(v)) + \sum_{c\in\mathcal{C}} c^{\mathrm{r}}_{m(c)}$$
$$\operatorname{duties}(S) := |\{v \in \mathcal{V} \mid d(v) \neq \emptyset\}|.$$

In summary, we formalize our problem as follows: Find a feasible schedule according to Definition 5 such that $\operatorname{cost}(S)$ as defined in Definition 6 is minimized.

Additional Assumptions

In the following, we summarize all the assumptions we make on the input data.
All costs are non-negative:

$$c^{\mathrm{v}} \geq 0 \qquad c^{\mathrm{d}}_{s,t} \geq 0 \qquad c^{\mathrm{t}}_t \geq 0 \qquad c^{\mathrm{r}}_m \geq 0 \qquad \text{for all } s,t \in \mathcal{T}, m \in \mathcal{M} \tag{2.1}$$

The fuel consumption is non-negative, except for refueling:

$$f_t^{\mathrm{t}} \geq 0 \qquad f_r^{\mathrm{t}} \leq 0 \qquad \text{for all } t \in \mathcal{T}, r \in \mathcal{R} \tag{2.2}$$

$$f_{s_1,s_2}^{\mathrm{d}} \geq 0 \qquad \text{for all } s_1 \in \mathcal{V} \cup \mathcal{T} \cup \mathcal{R}, s_2 \in \mathcal{T} \cup \mathcal{R} \tag{2.3}$$

There are no zero-time rentals:

$$z_t^{\mathrm{start}} < z_t^{\mathrm{end}} \qquad \text{for all } t \in \mathcal{T} \tag{2.4}$$

We assume the Triangle Inequalities for time, fuel consumption and cost. For $s \in \mathcal{V} \cup \mathcal{T} \cup \mathcal{R}$ and $r, t \in \mathcal{T} \cup \mathcal{R}$ holds:

$$t_{s,t} \leq t_{s,r} + t_{r,t} \qquad c_{s,t}^{\mathrm{d}} \leq c_{s,r}^{\mathrm{d}} + c_{r,t}^{\mathrm{d}} \qquad f_{s,t}^{\mathrm{d}} \leq f_{s,r}^{\mathrm{d}} + f_{r,t}^{\mathrm{d}} \tag{2.5}$$

From (2.1) - (2.5) we get for all $s, r, t \in \mathcal{T}$:

$$t_{s,t} \leq t_{s,r} + \left(z_r^{\mathrm{end}} - z_r^{\mathrm{start}}\right) + t_{r,t} \tag{2.6}$$

$$f_{s,t}^{\mathrm{d}} \leq f_{s,r}^{\mathrm{d}} + f_r^{\mathrm{t}} + f_{r,t}^{\mathrm{d}} \tag{2.7}$$

$$c_{s,t}^{\mathrm{d}} \leq c_{s,r}^{\mathrm{d}} + c_r^{\mathrm{t}} + c_{r,t}^{\mathrm{d}} \tag{2.8}$$

2.3 Classification

We aim to classify our problem in relation to other known problems in the literature and state the difficulty of these problems.

Vehicle Scheduling Problems

According to the structure of the problem stated in Section 2.2, we regard the field of vehicle scheduling problems (VSP). [BK09] define the VSP as follows: „Given a set of timetabled trips with fixed travel (departure and arrival) times and start and end locations as well as traveling times between all pairs of end stations, the objective is to find an assignment of trips to vehicles such that each trip is covered exactly once, each vehicle performs a feasible sequence of trips and the overall costs are minimized.“ The complexity of some variants of the VSP is regarded by [LK81].
A similar problem formulation is the dial-a-ride problem (DARP). [CL07] discuss the differences of the DARP to other vehicle routing problems and write: „What makes the DARP different from most such routing problems is the human perspective. When transporting passengers, reducing user inconvenience must be balanced against minimizing operating costs.“ The basic formulations of VSP and DARP are the same, therefore we use the formulation of VSP as it is more common.

Depot Variants

In [BK09], there are two main variants for the VSP with respect to where vehicles start and return to. In the single depot case (SD-VSP), there is one depot from where all vehicles start. After usage, all vehicles return to this depot. The multiple depot case (MD-VSP) means that there is more than one depot and from each depot a certain number of vehicles starts. After usage, each vehicle returns to the depot from where it has started.

In order to make our problem more realistic, we model more than one depot. There are more than one vehicle where each vehicle starts at its specific start position. The vehicles do not have a certain point where they have to return to after usage, i. e. they can stay wherever the last trip of their duty ends. [DP95] claim that „if the vehicle[s] are allowed to return to a depot different from its origin depot, [...] the problem can be solved as a single depot instance.“ We see that our problem is in the single depot case SD-VSP concerning the depot variant.

In [DF54], it is proven that SD-VSP can be solved in polynomial time, i. e. SD-VSP is in $\mathcal{P}$. In contrast, the multiple depot case MD-VSP is $\mathcal{NP}$-hard as shown in [BCG87].

Fuel Constraints

We further consider fuel constraints in our problem. The literature (cf. [BK09], [Raf83]) names general resources like time, mileage or fuel, summarized in the general term „route constraints“. The respective problems with route constraints are called SD-VSP-RC and MD-VSP-RC. [FP95] describe the VSP with time constraints, [Raf83] present the VSP with path constraints, which is a more general formulation. In these models, a vehicle returns to the depot after the respective resource is exhausted, while the vehicle has the possibility to refuel at certain locations in our model. The problem with not refilling the resource is a special case of the problem with the possibility to refill the resource. We see in Section 2.4 that SD-VSP-RC is already $\mathcal{NP}$-hard.

There are two approaches of the VSP including refueling stations in the literature, namely the Alternative Fuel VSP (AF-VSP) introduced by [Adl14] and the Electric VSP (E-VSP) introduced by [WLR$^+$16]. The E-VSP is defined as a „Multi-Depot VSP with distance constraints and charging possibilities. [...] Each vehicle can be recharged fully or partially at any given recharging station.“ [WLR$^+$16, p. 73]. If we identify the distance constraints with the fuel constraints, this formulation comes close to our problem setting. The differences to the E-VSP are that we regard the single-depot case and add cover constraints. The AF-VSP further only allows full recharging and assumes a constant charging time, where partial recharging is possible in our problem and the charging time depends on the remaining fuel state.

Cover Constraints

In the basic SD-VSP, all existing trips have to be fulfilled. In the underlying master theses, only a selection of the trips has to be fulfilled which provides a more general setting. Namely, there are „customers with sets of alternative trips out of which exactly one trip shall be fulfilled, respectively.“ ([Kai16, p. 10], [Kno16, p. 10]) In our problem, even more general cover constraints are required. There are customers with sets of alternative routes, consisting of trips; for each customer, exactly one route has to be fulfilled, i. e. each of its trips is fulfilled. We call these constraints „multi-leg“ cover constraints and write the problem VSP-MC. This is a generalization to the previous cover constraints, as can be seen easily by rewriting them: There are customers with sets of alternative routes, where each route consists of exactly one trip. According to this reformulation, we call the primary constraints „single-leg“ cover constraints and write the problem VSP-SC. We see in Section 2.4 that VSP-SC is already $\mathcal{NP}$-hard.

Conclusion

In summary, we have a problem with two types of constraints which individually make the problem $\mathcal{NP}$-hard, namely the multi-leg cover constraint and the fuel constraint. The only difference to [Kai16] and [Kno16] is that the single-leg cover constraint is replaced by the multi-leg cover constraint. Their solution methods are extended to our requirements in this thesis. To the knowledge of the author, these cover constraints have not been treated in the literature. Further, there is no appearance of cover constraints in general in combination with vehicle scheduling in the literature, besides the underlying theses.

2.4 Complexity

We regard the complexity of our problem. As we have seen in Section 2.3, the problem can be modeled as a single depot vehicle scheduling problem SD-VSP with resource constraints, the possibility of refueling and multi-leg cover constraints. The SD-VSP itself can be solved in polynomial time, which is proven by [DF54]. It can be formulated as a minimum-cost flow problem. If we extend the basic formulation with one of the additional constraints, it becomes $\mathcal{NP}$-hard.

Theorem 1 (VSP with resource constraints)**.** *The vehicle scheduling problem with resource constraints and the special objective of minimizing the number of used vehicles is $\mathcal{NP}$-hard.*

This theorem is proven in [Kai16, p. 11] and[Kno16, p. 11] by a polynomial reduction of the bin packing problem. This theorem holds for the case of resource constraints without the possibility of refilling the resources. Since this is a special case of resource constraints with refueling, this problem is also $\mathcal{NP}$-hard.

Theorem 2 (VSP with cover constraints)**.** *The vehicle scheduling problem with cover constraints and the special objective of minimizing the number of used vehicles is $\mathcal{NP}$-hard.*

This theorem is proven in [Kai16, p. 12] and [Kno16, p. 12] by a polynomial reduction of the set cover problem. This theorem treats the case of single-leg cover constraints.

Theorem 3 (VSP with multi-leg cover constraints)**.** *The vehicle scheduling problem with multi-leg cover constraints and the special objective of minimizing the number of used vehicles is $\mathcal{NP}$-hard.*

Proof. We prove this statement by a polynomial reduction of the VSP-SC. Consider a VSP-SC with vehicles $\hat{\mathcal{V}}$, customers $\hat{\mathcal{C}}$, trips $\hat{\mathcal{T}}$ and the function $\hat{C} : \hat{\mathcal{T}} \to \hat{\mathcal{C}}$ that maps trips to their respective customers. The problem is defined in detail in [Kai16] and [Kno16]. We create the corresponding VSP-MC as follows: $\mathcal{V} := \hat{\mathcal{V}}$, $\mathcal{C} := \hat{\mathcal{C}}$, $\mathcal{T} := \hat{\mathcal{T}}$ stay the same. We define the multimodal routes as one-element sequences for each trip

$$\mathcal{M} := \left\{(t) \mid t \in \hat{\mathcal{T}}\right\}$$

and the mappings

$$M : \mathcal{T} \to \mathcal{M},\ M(t) = (t), \qquad C : \mathcal{M} \to \mathcal{C},\ C(m) = \hat{C}(t) \qquad \text{for } t \in m \text{ unique.}$$

The solution of VSP-SC can easily be mapped to a solution of the corresponding VSP-MC and vice versa. If a trip is chosen in VSP-SC, the respective route is chosen in VSP-MC. Then, every trip of this route is fulfilled and the customer is satisfied. The other way round, if a route is chosen in VSP-MC, the trip contained in this route is chosen in VSP-SC and the customer is satisfied. The feasibility of the vehicle duties is not affected by this procedure.
This is a polynomial reduction, VSP-SC is $\mathcal{NP}$-hard by Theorem 2 and hence VSP-MC is $\mathcal{NP}$-hard. □

With Theorem 1 and Theorem 3 we can see that our problem gets $\mathcal{NP}$-hard only with cover constraints or with resource constraints and considering the number of vehicles. The problem becomes even harder, as we do not only consider the number of vehicles but include the operational cost and penalty terms for customer preferences. Further, we want to have the possibility to refuel during the process, i. e. we have negative resource cost. Finally, we aim to apply both of these constraints simultaneously.
We will model our problem as a mixed-integer linear program in a size polynomial in the input size. Therefore, it is possible to verify a solution in polynomial time. This means our problem is $\mathcal{NP}$-complete.

2.5 Approaches in the Literature

In the previous sections we have seen that the problem we are dealing with is very hard. We cannot expect to find an algorithm that solves the problem optimally running in polynomial time. Therefore we focus on the field of heuristical solution methods. Using them it is possible to find a good solution for realistic instances in reasonable time. The field of the vehicle scheduling problems is highly researched and there are many solution approaches available in the literature. [BK09] provide an extensive overview of the solution approaches for both problems SD-VSP and MD-VSP. We classify our problem as a SD-VSP with resource constraints and multi-leg cover constrains. This is only a small extension to the problem treated in the underlying theses where single-leg cover constraints are contained. As [Kai16] and [Kno16] already discuss the concerning approaches in the literature, we refer to them for more details. In the following, we present outlines of the solution methods for the AF-VSP and the E-VSP.

Heuristical Approaches

In order to tackle the AF-VSP, [Adl14] present a Greedy heuristic. The trips are iteratively added to the vehicle duty that gives the lowest cost-increasing provided that the duty stays feasible. During the feasibility check, also the respective refuel points are inserted to the duties. A new vehicle duty is created if no feasible appending is possible. This heuristic is fast but does not provide good results. Therefore, it is only appropriate for computing an initial solution.
[WLR+16] provide an Adaptive Large Neighborhood Search heuristic for the E-VSP. We present a rough outline of this algorithm. First, an initial solution is created by a greedy heuristic. From this solution, a certain number of trips is deleted (destroy method). After this, several new feasible duties are created by inserting respective trips with lowest cost-increasing at certain positions (repair method). Finally, the new duties are selected with minimal total cost, such that the duties cover each trip and the depot constraints are fulfilled. Several destroy and repair methods are presented there as well as a deterministic and a randomized version of the heuristic.

Optimal Approach

For an optimal solution of the AF-VSP, [Adl14] develop a Branch-and-Price procedure. A path flow formulation is considered which yields a master problem defining a set partitioning problems. The relaxed version of this master problem is solved via column generation. The subproblems for finding new feasible duties is a Weight Constrained Shortest Path Problem with Replenishment. As branching decisions, they use the fact if a certain trip is directly fulfilled after another or not. They use a Greedy heuristic for an initial solution and compute an optimal solution with this Branch-and-Price method.

Modeling the Refuel Points

There is no straightforward procedure to model the refuel points. It is not possible to use them individually in a task graph since the single vehicle duties cannot be distinguished if several duties visit the same refuel point. We tackle this issue by creating copies of the refuel points.
[Adl14] create a refuel point node after each depot and after each trip for each refuel point. These nodes represent the respective visit of the refuel point at the beginning of a duty and directly after a trip. It is not necessary to specify the charging time at the refuel point since the recharging time is assumed constant there. If we identify the depots with the vehicles in our problem, we have $(|\mathcal{V}| + |\mathcal{T}|)\,|\mathcal{R}|$ refuel point copies in the task graph. As we do not assume a constant charging time, we cannot directly apply this approach to our problem.
[WLR+16] create two copies for each refuel point and trip. One refuel point node is set between the depots and the trip, the other one is set after the trip. With this, they guarantee that a vehicle is able to refuel before the trip if the duty starts with this trip and after the trip. If we identify the depots with the vehicles in our problem, there are $2|\mathcal{T}||\mathcal{R}|$ refuel point copies in the task graph. Since partial refueling is allowed there, it is necessary to determine the respective refueling times. This is realized by an additional time variable in the formulation. In order to maintain this variable, a complete graph is necessary. Thus, a preprocessing on the graph, which eliminates edges between time-infeasible trips and is common usage for the VSP, is not possible there.
In the underlying theses, [Kai16] and [Kno16] create a copy between each feasible pair of trips for each refuel point. With this method it is possible to determine the respective charging time individually in advance and modeling partial refueling is easy. In the task graph, time-infeasible arcs are removed in advance and are not treated by additional constraints. In the worst case, there are $\left(\frac{1}{2}|\mathcal{T}|^2 + |\mathcal{V}||\mathcal{T}|\right)|\mathcal{R}|$ refuel point copies in the task graph. In order to reduce the size of the task graph, a preprocessing on the refuel point nodes based on Pareto-optimality is provided. Using this for realistic instances, most of the copies can be eliminated without cutting the optimal solution.

Applications

Besides car sharing with autonomous vehicles, there are further applications of this kind of problem in the literature, namely the routing of buses or the transport of cargo (cf. [WLR$^+$16], [Adl14]). The VSP in general is suitable if the time in which the respective customers are served is important. The VSP with fuel constraints is particularly applied for vehicles with limited fuel capacity and a scarce number of available refuel points, such as electric vehicles. The cover constraints can be used if the serving of a subset of the requests suffices, for example if customers have several alternatives. To the knowledge of the author, this issue has not been treated in the literature.

Chapter 3

Mathematical Models

We introduce the mathematical model, with which we want to solve the previously described problem. We define the underlying task graph and develop then an arc flow formulation on this graph. The main idea is to model a flow on the vehicles and the trips. This flow has to fulfill additional requirements such that the cover constraints and the fuel constraints are fulfilled. As mentioned before, the public transport trips are not part of the input. We therefore consider only the car trips in the mathematical model.

3.1 Task Graph

We introduce the task graph, on which the model is based. It is a directed graph based on the relation $\prec$ which we defined in Section 2.2, i. e. there is an edge (s,t) in the graph if $s \prec t$ holds. The graph is basically the same as used in [Kai16] and [Kno16] with the only difference that the customer and route considerations are adapted here.

Definition 7 (Task Graph)**.** Let $d^{\mathrm{s}}, d^{\mathrm{e}}$ be special vertices describing the source and sink of the vehicle flow. We define the task graph as $G = (V, A)$, where

$$V := \{d^{\mathrm{s}}, d^{\mathrm{e}}\} \mathbin{\dot\cup} \mathcal{V} \mathbin{\dot\cup} \mathcal{T}$$

is the vertex set consisting of the source, the sink, the vehicle set $\mathcal{V}$ and the trip set $\mathcal{T}$. The arc set is

$$A := (\{d^{\mathrm{s}}\} \times \mathcal{V}) \mathbin{\dot\cup} \left\{(s,t) \in (\mathcal{V} \mathbin{\dot\cup} \mathcal{T})^2 \,|\, s \prec t\right\} \mathbin{\dot\cup} ((\mathcal{V} \mathbin{\dot\cup} \mathcal{T}) \times \{d^{\mathrm{e}}\}) \,.$$

A vertex $s \in \mathcal{V}$ represents the initial state of a vehicle s where it becomes available for the first time. Each d^{s}-d^{e}-path in G is the duty of one vehicle, i.e. this vehicle fulfills the trips in the order given by the path. Hence, two trips are connected only if it is possible that one car fulfills both trips, i.e. the relation $\prec$ holds.

Lemma 1 (cf. [Kai16], [Kno16])**.** *G is a directed acyclic graph.*

Proof. Assume there is a cycle in G. The source d^{s} and sink d^{e} have only ingoing, respectively outgoing arcs and are therefore not part of the cycle. For $v \in \mathcal{V}$, all ingoing arcs come from d^{s}, hence $v \in \mathcal{V}$ are neither part of the cycle. This means, a cycle consists only of trips.
Consider an arbitrary cycle of trips $t_1, \dots, t_k \in \mathcal{T}$, $k \geq 2$. These trips form a cycle, i. e. $t_1 \prec \dots \prec t_k$ and $t_k \prec t_1$. With Definition 4 and the assumptions (2.4) and (2.5) holds:

$$\begin{aligned} & z_i^{\text{start}} < z_i^{\text{end}} \leq z_i^{\text{end}} + t_{t_i,t_{i+1}} \leq z_{i+1}^{\text{start}} < z_{i+1}^{\text{end}} && \text{for all } i \in [k-1] \\ \Rightarrow \quad & z_1^{\text{start}} < z_k^{\text{end}} \quad \Rightarrow \quad z_k^{\text{end}} + t_{t_1,t_k} > z_1^{\text{start}} && \Rightarrow \quad t_k \not\prec t_1 \end{aligned}$$

This is a contradiction. Therefore no cycle exists. □

In order to consider refueling and refuel stations, we introduce an extended task graph.

Definition 8 (Extended Task Graph)**.** For every $s, t \in \mathcal{V} \cup \mathcal{T}$ with $s \prec t$ we create a copy of $\left\{ r \in \mathcal{R} | z_s^{\text{end}} + t_{s,r} + t_{r,t} \leq z_t^{\text{start}} \right\}$ denoted by $\mathcal{R}_{s,t}$. This means, various copied sets are pairwise disjoint. The expression $r \in \mathcal{R}_{s,t}$ means that a vehicle is able to finish trip s, then drive to refuel station r and then start trip t in time.
We define the extended task graph $\widehat{G} = \left(\widehat{V}, \widehat{A}\right)$ with vertex set

$$\widehat{V} := V \cup \bigcup_{\substack{s,t \in \mathcal{V} \cup \mathcal{T} \\ s \prec t}} \mathcal{R}_{s,t}$$

and arc set

$$\widehat{A} := A \cup \{(s,r) | s,t \in \mathcal{V} \cup \mathcal{T}, s \prec t, r \in \mathcal{R}_{s,t}\} \cup \{(r,t) | s,t \in \mathcal{V} \cup \mathcal{T}, s \prec t, r \in \mathcal{R}_{s,t}\}.$$

It is possible that there is a copy of each refuel station for each feasible pair of trips. This leads to an enormous size of the task graph and thus a bad solution behavior is expected. [Kai16] and [Kno16] describe a method to reduce the size of $\mathcal{R}_{s,t}$ without cutting the optimal solution. This method only considers Pareto-optimal refuel stations w. r. t. a suitable function. From now on, we will use $\widehat{G} = \left(\widehat{V}, \widehat{A}\right)$ with restricted $\mathcal{R}_{s,t}$.

Lemma 2 (cf. [Kai16], [Kno16])**.** *$\widehat{G}$ is a directed acyclic graph.*

Proof. Assume there is a cycle in $\widehat{G}$. In comparison to G, only arcs (s,r) and (r,t) for $r \in \mathcal{R}_{s,t}, s \prec t$ were added. Assume there is a cycle containing $r \in \mathcal{R}_{s,t}$. r has only one ingoing arc (s,r) and one outgoing arc (r,t) and only if the arc (s,t) exists. There is no cycle on the vertices $\{s,r,t\}$. Every other cycle containing r is also a cycle using the arc (s,t). This is a contradiction to the fact that G is cycle-free as proven in Lemma 1. □

In the extended task graph $\widehat{G}$, a d^{s}-d^{e}-path further represents the duty of a vehicle. The additional arcs $(s, r), (r, t)$ for $r \in \mathcal{R}_{s,t}$ describe a possible detour between the trips s and t in order to refuel at refuel station r.
We introduce the following frequently used notation:

$$\mathrm{N}_G^-(t) := \{s \in V \mid (s,t) \in A\} \qquad \mathrm{N}_G^+(s) := \{t \in V \mid (s,t) \in A\}$$

$\mathrm{N}_G^-(t)$ is the set of in-neighbors of $t \in V$, $\mathrm{N}_G^+(s)$ is the set of out-neighbors of $s \in V$.

3.2 Arc Flow Formulation

In the following, we model the problem via a flow of the vehicles on the task graph. The trips and multimodal routes are given in advance. The fact whether two trips can be fulfilled subsequently in one duty, is already given by the underlying task graph. We additionally have to model the cover constraints and the fuel constraints. Since the duties of various vehicles are disjoint w. r. t. $\mathcal{T}$, we are able to use one common set of variables for the flow of the vehicles. From the flow, we can easily extract the individual d^{s}-d^{e}-paths in order to identify the duties of the respective vehicles.

Basic Model

We model the arc flow as a mixed-integer linear program. The formulation is basically built on the (MILP) formulation as described in ([Kai16, p. 34], [Kno16, p. 34]). We use the following decision variables:

- $x_{s,t} \in \{0, 1\}$ for $(s, t) \in A$:
  indicates, whether trip $t \in \mathcal{T}$ is fulfilled directly after $s \in \mathcal{V} \cup \mathcal{T}$
- $z_{s,r,t} \in \{0, 1\}$ for $t \in \mathcal{T}, s \in \mathrm{N}_G^-(t), r \in \mathcal{R}_{s,t}$:
  indicates, whether refuel station $r \in \mathcal{R}$ is visited between s and t
- $e_s \in [0, 1]$ for $s \in V \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}$:
  states the fuel of the respective vehicle after fulfilling trip $s \in \mathcal{T}$

If $s \in \mathcal{V}$, then $x_{s,t}$ determines whether trip t is the first trip fulfilled by s and e_s is the initial fuel state f_s^0 of vehicle s.
Additionally to (MILP), we introduce decision variables in order to ensure the cover constraints:

- $u_m \in \{0, 1\}$ for $m \in \mathcal{M}$: indicates whether multimodal route m is fulfilled

The basic constraints are developed by [Kai16] and [Kno16] and not explained in detail here. The basic constraints model a feasible flow of the vehicles, the refuel points and the fuel constraints. The cover constraints are developed now.

Cover Constraints

In (MILP), each customer has a set of alternative trips and from this set, exactly one trip has to be fulfilled. This is modeled as follows:

$$\sum_{t \in C^{-1}(c)} \sum_{s \in \mathrm{N}_G^-(t)} x_{s,t} = 1 \qquad \text{for all } c \in \mathcal{C} \tag{3.1}$$

In contrast to (MILP), here each customer has a set of alternative routes consisting of trips and from this set, exactly one route has to be fulfilled. Therefore, we replace (3.1) by the following formulation:

$$\sum_{m \in C^{-1}(c)} u_m = 1 \qquad \text{for all } c \in \mathcal{C} \tag{3.2}$$

$$\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t} = u_m \qquad \text{for all } m \in \mathcal{M}, t \in m \tag{3.3}$$

Constraint (3.2) ensures the completion of exactly one route of each customer. Constraint (3.3) says, if a route is fulfilled then every trip of this route must be fulfilled.

Objective Function

The objective function in (MILP) is given by

$$\sum_{s \in \mathcal{V}} \sum_{t \in \mathrm{N}_G^+(s) \setminus \{d^{\mathrm{e}}\}} x_{s,t} c^{\mathrm{v}} + \sum_{t \in \mathcal{T}} \sum_{s \in \mathrm{N}_G^-(t)} \left[x_{s,t} \left(c^{\mathrm{d}}_{s,t} + c^{\mathrm{t}}_t \right) + \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} \left(c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t} - c^{\mathrm{d}}_{s,t} \right) \right]$$

considering the vehicle costs c^{v}, the trip costs c^{t}_t for $t \in \mathcal{T}$ and the deadhead costs c^{d}. What is missing, are the route-dependent costs c^{r}. Thus, we add the term

$$\sum_{m \in \mathcal{M}} u_m c^{\mathrm{r}}_m$$

to the objective function.

LP Formulation

Putting all this together, we get the following formulation, called (MMILP):

$$
\begin{aligned}
\min \quad & \sum_{s\in\mathcal{V}} \sum_{t\in \mathrm{N}^+_G(s)\setminus\{d^{\mathrm{e}}\}} x_{s,t} c^{\mathrm{v}} + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m \\
& + \sum_{t\in\mathcal{T}} \sum_{s\in\mathrm{N}^-_G(t)} \left[x_{s,t}\left(c^{\mathrm{d}}_{s,t} + c^{\mathrm{t}}_t\right) + \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}\left(c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t} - c^{\mathrm{d}}_{s,t}\right)\right] && \text{(MMILP)} \\
\text{s.t.} \quad & \sum_{s\in\mathrm{N}^-_G(t)} x_{s,t} = \sum_{s\in\mathrm{N}^+_G(t)} x_{t,s} && \text{for all } t\in V\setminus\{d^{\mathrm{s}},d^{\mathrm{e}}\} \quad (3.4) \\
& \sum_{s\in\mathrm{N}^-_G(t)} x_{s,t} = 1 && \text{for all } t\in\mathcal{V} \quad (3.5) \\
& \sum_{m\in C^{-1}(c)} u_m = 1 && \text{for all } c\in\mathcal{C} \quad (3.2) \\
& \sum_{s\in\mathrm{N}^-_G(t)} x_{s,t} = u_m && \text{for all } m\in\mathcal{M}, t\in m \quad (3.3) \\
& \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} \le x_{s,t} && \text{for all } t\in\mathcal{T}, s\in\mathrm{N}^-_G(t) \quad (3.6) \\
& e_s \le f^0_s && \text{for all } s\in\mathcal{V} \quad (3.7) \\
& 0 \le e_s - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{s,r} && \text{for all } t\in\mathcal{T}, s\in\mathrm{N}^-_G(t) \quad (3.8) \\
& e_t \le 1 - f^{\mathrm{t}}_t - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{r,t} && \text{for all } t\in\mathcal{T}, s\in\mathrm{N}^-_G(t) \quad (3.9) \\
& e_t \le e_s - x_{s,t}\left(f^{\mathrm{d}}_{s,t} + f^{\mathrm{t}}_t\right) - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}\left(f^{\mathrm{d}}_{s,r} + f^{\mathrm{t}}_r + f^{\mathrm{d}}_{r,t} - f^{\mathrm{d}}_{s,t}\right) + (1 - x_{s,t}) \\
& && \text{for all } t\in\mathcal{T}, s\in\mathrm{N}^-_G(t) \quad (3.10) \\
& x_{s,t} \in \{0,1\} && \text{for all } (s,t)\in A \quad (3.11) \\
& z_{s,r,t} \in \{0,1\} && \text{for all } t\in\mathcal{T}, s\in\mathrm{N}^-_G(t), r\in\mathcal{R}_{s,t} \quad (3.12) \\
& e_s \in [0,1] && \text{for all } s\in V\setminus\{d^{\mathrm{s}},d^{\mathrm{e}}\} \quad (3.13) \\
& u_m \in \{0,1\} && \text{for all } m\in\mathcal{M} \quad (3.14)
\end{aligned}
$$

Model Equivalence

The problem formulation as developed in Chapter 3 is equivalent to to the problem setting as stated in Section 2.2. This means that each feasible schedule according to Definition 5 corresponds to a solution of the mixed-integer linear program (MMILP) and vice versa. This fact is shown in the following theorem.

Theorem 4 (Model Equivalence). *Let $S = (x, z, e, u)$ be a feasible solution of the (MMILP). We can transform S into a schedule S' that is feasible according to Definition 5. Let S' be a feasible schedule according to Definition 5. Then S' can be transformed into a feasible solution of the (MMILP). The objective value* $\operatorname{val}(S)$ *and the schedule cost* $\operatorname{cost}(S')$ *according to Definition 6 coincide.*

Proof. Let $S = (x, z, u, e)$ be a feasible solution of the (MMILP). We construct the schedule S' as follows: For each $v \in \mathcal{V}$, create a duty $d(v)$. Set $t_0 := v$. Then iteratively append (r_i, t_i) to the duty for the unique $t_i \in \mathcal{T}$ with $x_{t_{i-1},t_i} = 1$. Set $r_i := r$ for the unique $r \in \mathcal{R}_{t_{i-1},t_i}$ with $z_{t_{i-1},r,t_i} = 1$. Set $r_i := \emptyset$ if $z_{t_{i-1},r,t_i} = 0$ for all $r \in \mathcal{R}_{t_{i-1},t_i}$. The duty ends when $x_{t_i,d^{\mathrm{e}}} = 1$. If $x_{v,d^{\mathrm{e}}} = 1$ then we have $d(v) = \emptyset$. For each $c \in \mathcal{C}$ we set $m(c) := m$ for the unique $m \in C^{-1}(c)$ with $u_m = 1$. Due to the flow conservation (3.4) and the binary constraints (3.11) and (3.12), there is always a unique trip $t_i \in \mathcal{T}$ and at most one refuel point $r_i \in \mathcal{R}_{t_{i-1},t_i}$. The cover constraints (3.2) and the binary constraints (3.14) ensure that there is always a unique route $m \in C^{-1}(c)$.
Due to the construction of the graph G and the sets $\mathcal{R}_{s,t}$ for $s \in \mathcal{V} \cup \mathcal{T}$ with $s \prec t$, the time feasibility for S' is fulfilled. During a trip or deadhead trip, the fuel level of a vehicle decreases monotonously. While visiting a refuel points, the fuel level increases. Therefore we observe that a vehicle has its minimal fuel state directly before a refuel point or after a trip and its maximal fuel state directly after a refuel point or at the beginning of the duty. Constraints (3.8) and (3.13) ensure the fuel feasibility for the minimal fuel states and constraints (3.9) and (3.7) for the maximal fuel states. Constraint (3.10) describes the fuel consumption between two trips. The cover constraints are directly fulfilled by (3.2) and (3.3). Therefore S' is a feasible schedule according to Definition 5.
Let S' be a feasible schedule with $v \mapsto d(v)$ and $c \mapsto m(c)$. We construct a solution (x, z, u, e) as follows: Set $x_{d^{\mathrm{s}},v} := 1$ for all $v \in \mathcal{V}$. For each $v \in \mathcal{V}$, let $d(v) = ((r_1, t_1), \ldots, (r_{k_v}, t_{k_v}))$ and $t_0 := v$. For $i \in [k_v]$, set $x_{t_{i-1},t_i} := 1$ and $z_{t_{i-1},r_i,t_i} := 1$ if $r_i \neq \emptyset$. If $d(v) \neq \emptyset$ set $x_{t_{k_v},d^{\mathrm{e}}} := 1$, else set $x_{v,d^{\mathrm{e}}} := 1$. Set $e_v := f_v^0$ for $v \in \mathcal{V}$ and e_t as the respective fuel states after fulfilling trip $t \in \bigcup_{v \in \mathcal{V}} d(v)$. For each $c \in \mathcal{C}$, set $u_{m(c)} := 1$. All other variables that have not been mentioned, are set to 0. The proof of the feasibility works analogously to the first part of the proof.
The objective function of the (MMILP) is constructed as follows: The vehicle cost c^{v} arises for each non-empty duty. The deadhead cost between two subsequently fulfilled

trips s and t is $c^{\mathrm{d}}_{s,t}$ or $c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t}$ if refuel point $r \in \mathcal{R}_{s,t}$ is visited in-between. The trip and route costs are charged, iff the respective trip or route is fulfilled. Therefore holds

$$\mathrm{val}(S) = \mathrm{cost}\left(S'\right).$$

□

Remark 2. Consider that we transform a solution $S = (x, z, e, u)$ into S' and transform S' back to S'' as described in the proof. Note that the fuel states e_t of the solutions S and S'' do not necessarily coincide. This is caused by the construction of the (MMILP). The values for $x_{s,t}$, $z_{s,r,t}$ and u_m coincide and therefore the objective function is not affected.

Chapter 4

Successive Heuristics

In this chapter, successive heuristics are introduced in order to solve our problem. As seen in Section 2.4, the problem is $\mathcal{NP}$-hard even if we apply one of the restrictions, the cover constraints or the fuel constraints, individually. Our goal is to develop a heuristic that can cope with both multi-leg cover constraints and fuel constraints. We build our heuristic on a heuristic for a simpler version of the problem, developed in the underlying theses. [Kno16] present heuristical solution methods for the problem only with fuel constraints. The problem setting assumes that there is a set of trips where each of these trips shall be fulfilled. They already claim, that solving a complete instance of 24 hours to optimality is not possible with their respective computing capacity. Therefore it is a plausible assumption that an optimal solution for our problem cannot be expected in reasonable time.
Their solution methods are based on the idea of splitting the complete instance into several time intervals. For each interval, only the trips starting in the respective interval are considered. The connection between two trips from different intervals is redirected through a split point. From this formulation emerge several separate partial instances that are only loosely connected to each other via the split points. Each of these partial instances is solved separately and then the partial solutions are connected to a complete feasible solution. Two different approaches are presented in order to solve the problem: The constraints connecting the partial instances are relaxed by using Lagrange Relaxation. With suitable computation of Lagrange multipliers, the partial instances are solved in parallel. In the other approach, the partial instances are solved successively, where the respective connecting constraints are fixed beginning at the end.
An adaption of the cover constraints to the heuristic using Lagrange Relaxation seems not practicable. This heuristic heavily exploits the loose connection of the partial instances. The cover constraints strongly influence the complete instance by selecting the fulfilled trips, the multi-leg cover constraints even require an additional set of variables, belonging to none of the partial instances. Therefore, an additional relaxing of these cover constraints is not a promising approach. Instead, we focus on the second approach of Successive Heuristics.
The crucial difficulty for this procedure is to ensure the customer satisfaction. In

particular, if trips of a customer are wide apart in terms of time, these trips will lie in different splittings. This makes it hard to keep control over the trip selection in separately solved partial instances.
We first define the splitting of the instance and the arising adaptions of task graph and model. Then, we describe the heuristic in general. Finally, we introduce different splitting methods, one according to the customers and one according to time.

4.1 Successive Heuristics

4.1.1 Splitting the Problem

In order to create the partial instances, we define splittings of $\mathcal{T}$. In contrast to [Kno16], we define the splittings in a general way.

Definition 9 (Splitting)**.** Let $n \in \mathbb{N}$ and let

$$\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$$

be a partition of the set of trips. Then we call $\{\mathcal{T}_i \mid i \in [n]\}$ splitting of $\mathcal{T}$ and $\mathcal{T}_i$ partial trip set.

Adaption of the Task Graph

We transform our task graph such that it contains the splitting as defined in Definition 9. For this, we introduce so called split points connecting the partial sets. Arcs that connect two partial sets in the original formulation, take a detour over the respective split point in the transformed graph.

Definition 10 (Transformed Task Graph)**.** Let $\{\mathcal{T}_1, \dots \mathcal{T}_n\}$ be a splitting of $\mathcal{T}$ according to Definition 9. Then we define:

1. Split Point: Let $s \in \mathcal{T}_i$ for $i \in [n]\backslash\{1\}$. For $j \in [i-1]$, we define the split point $\mathrm{SP}_j(s)$ with $p^{\text{start}}_{\mathrm{SP}_j(s)} = p^{\text{end}}_{\mathrm{SP}_j(s)} =: p^{\text{start}}_s$, $z^{\text{start}}_{\mathrm{SP}_j(s)} = z^{\text{end}}_{\mathrm{SP}_j(s)} =: z^{\text{start}}_s$ and $f^{\text{t}}_{\mathrm{SP}_j(s)} =: 0$.
2. For $i \in [n]\backslash\{1\}$ and $j \in [i-1]$, we define $\mathcal{P}_{j,i} := \{\mathrm{SP}_j(s) \mid s \in \mathcal{T}_i\}$.
3. Partial Split Point Set: For $j \in [n-1]$, we define the partial split point set $\mathcal{P}_j := \bigcup_{i=j+1}^{n} \mathcal{P}_{j,i}$.
4. Split Point Set: We define the split point set $\mathcal{P} := \bigcup_{j=1}^{n-1} \mathcal{P}_j$.

Let $G = (V, A)$ be the task graph.

5. For $i \in [n], t \in \mathcal{T}_i$ and $j \in [i-1]$ we define $s \prec \mathrm{SP}_j(t) :\Leftrightarrow s \prec t$.

6. Transformed Task Graph: We define the transformed task graph $\overline{G} = \left(\overline{V}, \overline{A}\right)$ with vertex set

$$\overline{V} := V \cup \mathcal{P} = V \cup \{\mathrm{SP}_i(s) \mid i \in [n-1], j \in [n+1] \backslash [i], s \in \mathcal{T}_j\}$$

and arc set

$$\begin{aligned}\overline{A} := &(d^{\mathrm{s}} \times \mathcal{V}) \mathbin{\dot\cup} \{(s,t) \in (\mathcal{V} \mathbin{\dot\cup} \mathcal{T}_1) \times (\mathcal{T}_1 \mathbin{\dot\cup} \mathcal{P}_1) \mid s \prec t\} \\ &\mathbin{\dot\cup} \bigcup_{i=2}^{n} \{(s,t) \in \mathcal{T}_i \times (\mathcal{T}_i \mathbin{\dot\cup} \mathcal{P}_i) \mid s \prec t\} \\ &\mathbin{\dot\cup} \bigcup_{i=2}^{n} \left(\bigcup_{j=1}^{i-1} \{(s,t) \in \mathcal{P}_{j,i} \times \mathcal{T}_i \mid s = \mathrm{SP}_j(t)\} \right) \mathbin{\dot\cup} ((\mathcal{V} \mathbin{\dot\cup} \mathcal{T}) \times \{d^{\mathrm{e}}\}) .\end{aligned}$$

Adaption of the Model

In order to adapt the (MMILP) to the transformed task graph, we make the following considerations:
For all split points we define the costs and fuel states as

$$c^{\mathrm{t}}_s := 0 \qquad c^{\mathrm{d}}_{s,t} := 0 \qquad f^{\mathrm{t}}_s := 0 \qquad f^{\mathrm{d}}_{s,t} := 0 \qquad \text{for } s \in \mathcal{P}, t \in \mathrm{N}^{+}_{\overline{G}}(s)$$

since $p^{\mathrm{end}}_s = p^{\mathrm{start}}_t$ and $z^{\mathrm{end}}_s = z^{\mathrm{start}}_t$. Furthermore, refueling is not possible between s and t.
In the task graph, the arcs between two trips of different splittings are replaced by the detour over the splitting point. Therefore, the trip costs of a trip directly after a split point are not considered in the objective function any more. In order to compensate this, we add the following term to the objective function:

$$\sum_{s \in \mathcal{P}} \sum_{t \in \mathrm{N}^{+}_{\overline{G}}(s)} x_{s,t} c^{\mathrm{t}}_t$$

We want to ensure the flow conservation also in the new nodes $\mathcal{P}$, thus we add

$$\sum_{t \in \mathrm{N}^{-}_{\overline{G}}(s)} x_{t,s} = \sum_{t \in \mathrm{N}^{+}_{\overline{G}}(s)} x_{s,t} \qquad \text{for all } s \in \mathcal{P}. \tag{4.1}$$

The equations (3.4) and (4.1) are contracted to

$$\sum_{t \in \mathrm{N}^{-}_{\overline{G}}(s)} x_{t,s} = \sum_{t \in \mathrm{N}^{+}_{\overline{G}}(s)} x_{s,t} \qquad \text{for all } s \in \overline{V} \backslash \{d^{\mathrm{s}}, d^{\mathrm{e}}\} . \tag{4.2}$$

$$
\begin{aligned}
\min \quad & \sum_{s\in\mathcal{V}} \sum_{t\in \mathrm{N}^{+}_{\overline{G}}(s)\setminus\{d^{\mathrm{e}}\}} x_{s,t}c^{\mathrm{v}} + \sum_{s\in\mathcal{P}} \sum_{t\in \mathrm{N}^{+}_{\overline{G}}(s)} x_{s,t}c^{\mathrm{t}}_{t} + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_{m} \\
& + \sum_{t\in\mathcal{T}\cup\mathcal{P}} \sum_{s\in \mathrm{N}^{-}_{\overline{G}}(t)\setminus\mathcal{P}} \left[x_{s,t}\left(c^{\mathrm{d}}_{s,t}+c^{\mathrm{t}}_{t}\right) + \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}\left(c^{\mathrm{d}}_{s,r}+c^{\mathrm{d}}_{r,t}-c^{\mathrm{d}}_{s,t}\right)\right] && \text{(SMILP)}\\
\text{s.t.} \quad & \sum_{t\in \mathrm{N}^{-}_{\overline{G}}(s)} x_{t,s} = \sum_{t\in \mathrm{N}^{+}_{\overline{G}}(s)} x_{s,t} && \text{for all } s\in\overline{V}\setminus\{d^{\mathrm{s}},d^{\mathrm{e}}\} && (4.2)\\
& \sum_{s\in \mathrm{N}^{-}_{\overline{G}}(t)} x_{s,t} = 1 && \text{for all } t\in\mathcal{V} && (3.5)\\
& \sum_{m\in C^{-1}(c)} u_m = 1 && \text{for all } c\in\mathcal{C} && (3.2)\\
& \sum_{s\in \mathrm{N}^{-}_{\overline{G}}(t)} x_{s,t} = u_m && \text{for all } m\in\mathcal{M}, t\in m && (3.3)\\
& \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} \le x_{s,t} && \text{for all } t\in\mathcal{T}\cup\mathcal{P}, s\in \mathrm{N}^{-}_{\overline{G}}(t)\setminus\mathcal{P} && (4.3)\\
& e_s \le f^{0}_{s} && \text{for all } s\in\mathcal{V} && (3.7)\\
& 0 \le e_s - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}f^{\mathrm{d}}_{s,r} && \text{for all } t\in\mathcal{T}\cup\mathcal{P}, s\in \mathrm{N}^{-}_{\overline{G}}(t)\setminus\mathcal{P} && (4.4)\\
& e_t \le 1 - f^{\mathrm{t}}_{t} - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}f^{\mathrm{d}}_{r,t} && \text{for all } t\in\mathcal{T}\cup\mathcal{P}, s\in \mathrm{N}^{-}_{\overline{G}}(t)\setminus\mathcal{P} && (4.5)\\
& e_t \le e_s - x_{s,t}\left(f^{\mathrm{d}}_{s,t}+f^{\mathrm{t}}_{t}\right) - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}\left(f^{\mathrm{d}}_{s,r}+f^{\mathrm{t}}_{r}+f^{\mathrm{d}}_{r,t}-f^{\mathrm{d}}_{s,t}\right) + (1-x_{s,t}) \\
& && \text{for all } t\in\mathcal{T}\cup\mathcal{P}, s\in \mathrm{N}^{-}_{\overline{G}}(t)\setminus\mathcal{P} && (4.6)\\
& e_t \le e_s - x_{s,t}f^{\mathrm{t}}_{t} + (1-x_{s,t}) && \text{for all } s\in\mathcal{P}, t\in \mathrm{N}^{+}_{\overline{G}}(s) && (4.10)\\
& x_{s,t}\in\{0,1\} && \text{for all } (s,t)\in\overline{A} && (4.7)\\
& z_{s,r,t}\in\{0,1\} && \text{for all } t\in\mathcal{T}\cup\mathcal{P}, s\in \mathrm{N}^{-}_{\overline{G}}(t)\setminus\mathcal{P}, r\in\mathcal{R}_{s,t} && (4.8)\\
& e_s\in[0,1] && \text{for all } s\in\overline{V}\setminus\{d^{\mathrm{s}},d^{\mathrm{e}}\} && (4.9)\\
& u_m\in\{0,1\} && \text{for all } m\in\mathcal{M} && (3.14)
\end{aligned}
$$

The fuel constraints are adapted in the following way: (3.6), (3.8), (3.9) and (3.10) hold also on the arcs leading to $\mathcal{P}$ and are therefore replaced by (4.3), (4.4), (4.5) and (4.6).
Further the arcs leading from a split points to its respective trips have to be considered. Since refueling is not possible there, we have only to adapt (3.10). Since $f^{\mathrm{d}}_{s,t} = 0$ and

refueling is not possible between s and t, the constraint reads as follows:

$$e_t \leq e_s - x_{s,t} f_t^{\mathrm{t}} + (1 - x_{s,t}) \qquad \text{for all } s \in \mathcal{P}, t \in \mathrm{N}_G^+(s) \tag{4.10}$$

The customer constraints (3.2) are not affected by transforming the graph. The decision whether a trip $t \in \mathcal{T}$ is fulfilled is still given by $\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}$, no matter if the ingoing arc is a split point or not. Thus, the route constraints (3.3) do not change either.
Putting all together, we have the formulation (SMILP).

Model Equivalence

We show that (SMILP) is also feasible in (MMILP). Inversely, not every solution of the original formulation is feasible in the split formulation.

Theorem 5 (Transformation of (SMILP) into (MMILP))**.** *Let $S = (x, z, e, u)$ be a feasible solution of the* (SMILP)*. Then S can be converted to a solution $\bar{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ which is feasible in the* (MMILP)*. The objective values* $\mathrm{val}(S)$ *and* $\mathrm{val}(\bar{S})$ *coincide.*

Proof. Based on a feasible solution of the split formulation (SMILP), we construct a feasible solution of the original formulation (MMILP). We examine how the trip connections via split points correspond to the trip connections without split points and the behavior of the refuel points during these connections. Finally, we show that the constructed solution is feasible in the original formulation.
Let $S = (x, z, e, u)$ be a feasible solution of the (SMILP). We define $\bar{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ as follows: Let $s \in \mathcal{V}$, $t \in \mathcal{T}$ with $s \prec t$ and $r \in \mathcal{R}_{s,t}$. We set:

$$\bar{x}_{s,t} := \begin{cases} x_{s,t} & \text{if } t \in \mathcal{T}_1 \\ x_{s,\mathrm{SP}_1(t)} & \text{otherwise} \end{cases} \qquad \bar{z}_{s,r,t} := \begin{cases} z_{s,r,t} & \text{if } t \in \mathcal{T}_1 \\ z_{s,r,\mathrm{SP}_1(t)} & \text{otherwise} \end{cases}$$

For $i \in [n], s \in \mathcal{T}_i$, $t \in \mathcal{T}$ with $s \prec t$ and $r \in \mathcal{R}_{s,t}$, we set:

$$\bar{x}_{s,t} := \begin{cases} x_{s,t} & \text{if } t \in \mathcal{T}_i \\ x_{s,\mathrm{SP}_i(t)} & \text{if } t \in \bigcup_{j=i+1}^n \mathcal{T}_j \\ 0 & \text{otherwise} \end{cases} \qquad \bar{z}_{s,r,t} := \begin{cases} z_{s,r,t} & \text{if } t \in \mathcal{T}_i \\ z_{s,r,\mathrm{SP}_i(t)} & \text{if } t \in \bigcup_{j=i+1}^n \mathcal{T}_j \\ 0 & \text{otherwise} \end{cases}$$

Additionally, we set $\bar{e}_s := e_s$ for all $s \in \mathcal{V} \,\dot\cup\, \mathcal{T}$ and $\bar{u}_m := u_m$ for all $m \in \mathcal{M}$. From the flow conservation (4.2) and $\mathrm{N}_G^+(\mathrm{SP}_i(t)) = \{t\}$ follows:

$$x_{s,\mathrm{SP}_i(t)} = 1 \Rightarrow x_{\mathrm{SP}_i(t),t} = 1 \qquad \text{for } i \in [n], s \in \mathcal{T}_i, t \in \bigcup_{j=i+1}^n \mathcal{T}_j \text{ with } s \prec t$$

The time feasibility of $\bar{S}$ follows directly from the definition of $s \prec \mathrm{SP}_i(t)$. Visiting a refuel point between s and $\mathrm{SP}_i(t)$ in the (SMILP) is equivalent to visiting a refuel point between s and t in the (MMILP). Visiting a refuel point between $\mathrm{SP}_i(t)$ and t is not feasible in the (SMILP). The fuel constraints are equivalent by construction and therefore $\bar{S}$ is feasible w. r. t. fuel. The cover constraints are equivalent in both formulations. Therefore $\bar{S}$ is a feasible solution of the (MMILP).
The objective function of (SMILP) is adapted such that the trip and deadhead costs arise for exactly the same trips, the vehicle and route costs are not affected by the transformation. Thus $\mathrm{val}(S) = \mathrm{val}(\bar{S})$. □

Remark 3. Note that a feasible solution in (MMILP) is not necessarily feasible in (SMILP). If there are $s, t \in \mathcal{T}$ with $s \prec t$ and $s \in \mathcal{T}_{i+1}, t \in \mathcal{T}_i$, then s and t cannot be connected in the transformed task graph. This issue is further discussed in Section 4.2.

Example

In the following, we show an example how the transformed task graph is created for a given splitting. We do not enlarge upon the kind of the splitting, as this is treated in Section 4.2 and Section 4.3. Figure 4.1 shows a simplified model of the transformed task graph $\overline{G}$ with $n = 3$. The boxes $\{I_1, I_2, I_3\}$ indicate how the complete instance is split up in partial instances. This is needed later in order to define the heuristics.

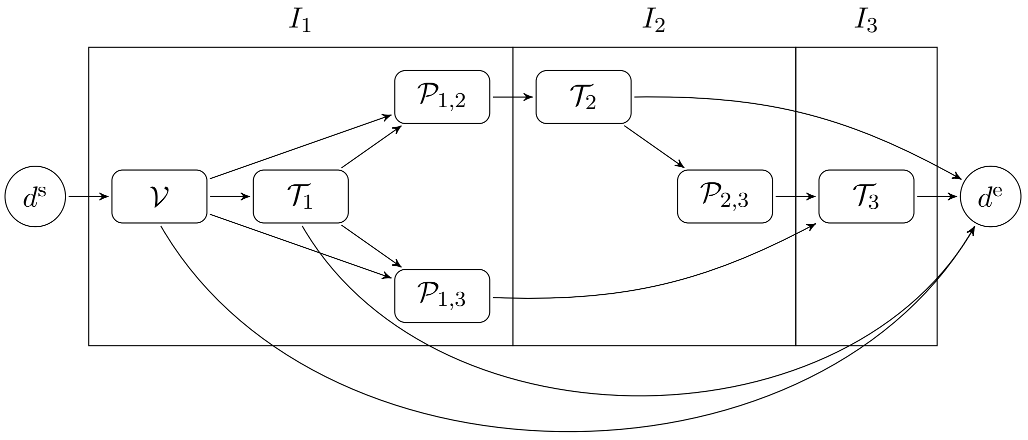

Figure 4.1: Simplified representation of $\overline{G}$ for $n = 3$

We show an exemplary instance, with which we explain the creation of the split points.

Example 1. Let $\mathcal{V} = \{v_1, v_2, v_3\}$ and $\mathcal{T} = \{t_1, \dots, t_6\}$. We assume that each trip belongs to a different customer, therefore the cover constraints are equivalent to fulfilling each trip. We consider a splitting

$$\mathcal{T}_1 = \{t_1\} \qquad \mathcal{T}_2 = \{t_2, t_3\} \qquad \mathcal{T}_3 = \{t_4, t_5, t_6\}\,.$$

We assume that $s \prec t$ for $s \in \mathcal{T}_i, t \in \mathcal{T}_j$ with $i < j$. The partial split point sets are the following:

$$\mathcal{P}_{1,2} = \{\mathrm{SP}_1(t_2), \mathrm{SP}_1(t_3)\} \qquad \mathcal{P}_{1,3} = \{\mathrm{SP}_1(t_4), \mathrm{SP}_1(t_5), \mathrm{SP}_1(t_6)\}$$
$$\mathcal{P}_{2,3} = \{\mathrm{SP}_2(t_4), \mathrm{SP}_2(t_5), \mathrm{SP}_2(t_6)\}$$

For all $s \in \mathcal{T}_i$, $t \in \mathcal{T}_j$ with $i < j$, the arc (s, t) is replaced by the arc $(s, \mathrm{SP}_i(t))$ and the arc $\left(\mathrm{SP}_{j'}(t), t\right)$ is inserted for each $j' < j$.

4.1.2 General Setting

In this section, we describe the general setting of the heuristic. Let $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ be a splitting. For each partial trip set, we create a partial instance $I_1, \ldots, I_n$ which contains exactly the partial trip set $\mathcal{T}_i$ and some endpoints $\hat{\mathcal{P}}_i$. How these endpoints are created is explained afterwards. The partial instance I_1 additionally contains the vehicle set. The partial instances are solved from the end to the start, i. e. $I_n, \ldots, I_1$ are solved successively. For each partial instance I_i, a partial solution S_i is computed which is based on the already solved partial instances. In I_1 which is solved last, the partial duties are actually assigned to the vehicles. Finally the partial solutions are feasibly connected to an overall solution.
This heuristic is directly applied from [Kno16, Sec. 10.4]. As presented there, the order in which the partial instances are solved can be stated arbitrarily as long as I_1 is solved last. They argue that the setting with ordering from the end to the start suits best to the underlying instance structure. Therefore we present only this approach here.

Determination of the Endpoints

The sets of end points $\hat{\mathcal{P}}_i$ are initially empty for all $i \in [n]$. We first solve the partial instance I_n with $\hat{\mathcal{P}}_n = \emptyset$. For $i \in [n-1]$ assume that we have solved I_{i+1} right before. Based on the received partial solution S_{i+1}, we update the endpoint set for the preceding partial instance I_i.
Each duty of S_{i+1} consists of a sequence of trips out of $\mathcal{T}_{i+1}$ and refuel points and possibly ends with an end point out of $\hat{\mathcal{P}}_{i+1}$. It is possible that a duty only consists of an end point. For each duty we create an end point in $\hat{\mathcal{P}}_i$. If the duty starts with a trip or end point s, we create the end point t with the properties

$$p_t^{\text{start}} = p_t^{\text{end}} := p_s^{\text{start}} \qquad z_t^{\text{start}} = z_t^{\text{end}} := z_s^{\text{start}} \qquad f_t^0 := e_s + f_s^{\text{t}}$$

where e_s is the respective value of decision variable e in S_{i+1}. We add t to the end point set $\hat{\mathcal{P}}_i$.
For each end point $t \in \hat{\mathcal{P}}_i$, we call the respective trip $s \in \mathcal{T}_{i+1}$ from where it is created, the trip representing t. If an end point $t \in \hat{\mathcal{P}}_i$ is created from an end point $s \in \hat{\mathcal{P}} : i+1$, the trip representing t is the trip representing s. Note that each end point $t \in \hat{\mathcal{P}}_i$ has a trip $s \in \bigcup_{j=i+1}^{n} \mathcal{T}_j$ representing it.
The partial instance I_1 has a special role. This instance additionally contains the vehicle set. Each duty of the partial solution S_1 starts with a vehicle $v \in \mathcal{V}$, consists of trips out of $\mathcal{T}_1$ and refuel points and possibly ends with an endpoint out of $\hat{\mathcal{P}}_1$. It is possible that a duty only consists of a vehicle.

Feasible Connection of Partial Solutions

In order to generate an overall solution which is feasible for (MMILP), we connect the partial solutions. Let $\{S_1, \dots S_n\}$ be the partial solutions, solved as described before with the start and end points created as before. The connection works as follows:
For each duty in S_1, we check whether it ends with an end point $t \in \bigcup_{i=1}^{n} \hat{\mathcal{P}}_i$. We call this duty start duty.

- If it does, we delete the end point and append the duty of S_i for $i \in [n] \backslash \{1\}$ that starts with the trip representing t to the start duty. We then restart this procedure with the new end of the start duty.
- If it does not, the start duty is finished and we continue with the next duty in S_1.

In a partial instance I_i, each end point $t \in \hat{\mathcal{P}}_i$ is covered by some duty in S_i. This guarantees that all duties of the partial solutions are finally part of the overall solution. Algorithm 1 describes the procedure of the Successive Heuristics.

Example

In the following, we show an example how the end points are created and the partial solutions are connected. This example proceeds Example 1.
Example 2. Let $\mathcal{V} = \{v_1, v_2, v_3\}$ and $\mathcal{T} = \{t_1, \dots, t_6\}$ with the splitting $\{\{t_1\}, \{t_2, t_3\}, \{t_4, t_5, t_6\}\}$. We first solve the partial instance I_3 with end point set $\hat{\mathcal{P}}_2$ and partial solution S_3:

$$\hat{\mathcal{P}}_3 = \emptyset \qquad\qquad S_3 = \{(t_4, t_5), (t_6)\}$$

We create an end point t_4' out of t_4 and t_6' out of t_6. We solve the partial instance I_2 with end point set $\hat{\mathcal{P}}_2$ and partial solution S_2:

$$\hat{\mathcal{P}}_2 = \{t_4', t_6'\} \qquad\qquad S_2 = \{(t_4'), (t_2, t_3, t_6')\}$$

Algorithm 1: Successive Heuristic (general setting)

```
   Input: splitting $\{\mathcal{T}_1, \dots, \mathcal{T}_n\}$, vehicle set $\mathcal{V}$
   Output: overall solution $S$ with duty set $D$
1  foreach $i \in [n]$ do $\hat{\mathcal{P}}_i \leftarrow \emptyset$;
2  foreach $i = n, \dots, 2$ do
3      solve $I_i$, receive partial solution $S_i$ with duty set $D_i$;
4      foreach $D_i \ni d = (s_1, \dots, s_l)$ do
5          create end point $t$;
6          $p_t^{\text{start}} \leftarrow p_{s_1}^{\text{start}}, p_t^{\text{end}} \leftarrow p_{s_1}^{\text{start}}, z_t^{\text{start}} \leftarrow z_{s_1}^{\text{start}}, z_t^{\text{end}} \leftarrow z_{s_1}^{\text{start}}, f_t^0 \leftarrow e_{s_1} + f_{s_1}^{\text{t}}$;
7          $\hat{\mathcal{P}}_{i-1} \leftarrow \hat{\mathcal{P}}_{i-1} \cup \{t\}$;
8      end
9  end
10 solve $I_1$, receive partial solution $S_1$ with duty set $D_1$;
11 foreach $D_1 \ni d = (s_1, \dots, s_l)$ do
12     $t \leftarrow s_l$;
13     while $t \in \bigcup_{i=1}^{n} \hat{\mathcal{P}}_i$ do
14         determine duty $d' = (s_{l+1}, \dots, s_{l'})$ with $t_{l+1} \in \mathcal{T}$ representing $t$;
15         $d \leftarrow (s_1, \dots, s_{l-1}, s_{l+1}, \dots, s_{l'})$;
16         $t \leftarrow s_{l'}; l \leftarrow l' - 1$;
17     end
18 end
19 $D \leftarrow D_1$;
20 return $S$, $D$
```

We create an end point t_4'' out of t_4' and t_2' out of t_2. Note that t_4 represents t_4''. We solve the partial instance I_1 with end point set $\hat{\mathcal{P}}_1$ and partial solution S_1:

$$\hat{\mathcal{P}}_1 \, \{t_2', t_4''\} \qquad\qquad S_1 = \{(v_1, t_1, t_4''), (v_2, t_2'), (v_3)\}$$

Then we connect the partial solutions to

$$d(v_1) = (t_1, t_4, t_5) \qquad\qquad d(v_2) = (t_2, t_3, t_6) \qquad\qquad d(v_3) = \emptyset.$$

This is a feasible solution according to Definition 5.

4.1.3 Solving the Subproblems

We describe how the partial instances $I_1, \dots, I_n$ are solved. We create a task graph containing $\mathcal{T}_i$ and $\hat{\mathcal{P}}_i$ and solve this partial instance similar to the (SMILP). We first describe the procedure for $i \in [n] \backslash \{1\}$. Finally we explain the modifications to solve I_1.

Remark 4. The formulation developed here is only a basic structure for solving the partial instances. The realization of the cover constraints depends on the splitting method. Since there are several approaches to split the trip set, we refer to Section 4.2 and Section 4.3 for the actual description of the cover constraints. The route costs heavily depend on the cover constraints and are thus also neglected here. The approach as developed in Section 4.1 is not appropriate to gain meaningful results. Since the cover constraints are completely left out here, the empty solution is feasible and is obviously the cost-minimal solution.

Task Graph

First we define the task graph with which we can solve the partial instances. The transformed task graph $\overline{G}$ covers the complete instance, but contains already the partial trip sets of the splitting. We divide $\overline{G}$ into partial task graphs. For $i \in [n] \setminus \{1\}$, the partial task graph $\overline{G}_i$ contains the respective partial trip set $\mathcal{T}_i$ and the end point set $\hat{\mathcal{P}}_i$. The graph is defined as follows:

Definition 11 (Partial Transformed Task Graph)**.** Let $i \in [n] \setminus \{1\}$. For a set of end points $\hat{\mathcal{P}}_i$ and the partial trip set $\mathcal{T}_i$, the partial transformed task grah is the directed graph $\overline{G}_i = \left(\overline{V}_i, \overline{A}_i\right)$ with vertex set

$$\overline{V}_i := \{d^{\mathrm{s}}, d^{\mathrm{e}}\} \mathbin{\dot\cup} \mathcal{T}_i \mathbin{\dot\cup} \hat{\mathcal{P}}_i$$

and arc set

$$\overline{A}_i := \left(\{d^{\mathrm{s}}\} \times \left(\mathcal{T}_i \mathbin{\dot\cup} \hat{\mathcal{P}}_i\right)\right) \mathbin{\dot\cup} \left\{(s,t) \in \mathcal{T}_i \times \left(\mathcal{T}_i \mathbin{\dot\cup} \hat{\mathcal{P}}_i\right) \mid s \prec t\right\} \mathbin{\dot\cup} \left(\left(\mathcal{T}_i \mathbin{\dot\cup} \hat{\mathcal{P}}_i\right) \times \{d^{\mathrm{e}}\}\right).$$

In the partial instance are no vehicles. Thus each duty starts with a trip or end point.

Solving the Partial Instances

Let $i \in [n] \setminus \{1\}$. In order to solve each partial instance, we create a formulation which is based on the partial transformed task graph $\overline{G}_i$. The flow constraints and the fuel constraints are basically the same as in (SMILP), restricted to $\overline{G}_i$.
As mentioned before, there is no vehicle set any more. Instead all endpoints have to be visited. Therefore we replace (3.5) by

$$\sum_{s \in \mathrm{N}^-_{\overline{G}_i}(t)} x_{s,t} = 1 \qquad \text{for all } t \in \hat{\mathcal{P}}_i. \tag{4.11}$$

We are given initial fuel levels for the end points. They indicate the required fuel for the start of the next partial duty. These fuel levels work as lower bounds for the end of the duties in this partial instance. Therefore we introduce the constraints

$$f^0_s \leq e_s \qquad \text{for all } s \in \hat{\mathcal{P}}_i. \tag{4.12}$$

Since there are no vehicles in the partial instance, the constraint (3.7) is dropped.
We introduce two additional constraints: If a duty starts or ends with a trip, it is necessary that a vehicle can reach some refuel point before or after fulfilling the trip. Thus, if a duty starts or ends with a trip, the fuel at the start or at the end of this duty is bounded by $f^{\min}$ or $f^{\max}$, respectively. We define

$$f^{\min}_t := \min_{r\in\mathcal{R}} f^{\mathrm{d}}_{t,r} \qquad f^{\max}_t := 1 - \min_{r\in\mathcal{R}} f^{\mathrm{t}}_{r,t} \qquad \text{for } t \in \mathcal{T}.$$

Value $f^{\min}_t$ is the minimal fuel level at the end of $t \in \mathcal{T}$ such that the next refuel point can be reached afterwards. Value $f^{\max}_t$ is the maximal fuel state at the start of $t \in \mathcal{T}$ if some refuel point has been visited before. The constraints are the following:

$$e_s + f^{\mathrm{t}}_s \leq f^{\max}_s + (1 - x_{d^{\mathrm{s}},s}) \cdot \left(1 + f^{\mathrm{t}}_s\right) \qquad \text{for all } s \in \mathcal{T}_i \tag{4.13}$$

$$f^{\min}_s \leq e_s + (1 - x_{s,d^{\mathrm{e}}}) \qquad \text{for all } s \in \mathcal{T}_i \tag{4.14}$$

While solving the partial instances partial duties are created. For $i \in [n]\backslash\{1\}$, the number of these duties is not bounded so far. The number of duties in S_{i+1} equals $|\hat{\mathcal{P}}_i|$. If $|\hat{\mathcal{P}}_1| > |\mathcal{V}|$, the partial instance I_1 is infeasible and then the complete heuristic is infeasible. In order to prevent this, we restrict the number of created duties by the following inequality:

$$\sum_{t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i} x_{d^{\mathrm{s}},t} \leq |\mathcal{V}| \tag{4.15}$$

As mentioned before, it requires some additional work to include the cover constraints into the partial instances. The fulfilling of the cover constraints is also part of the respective heuristic and therefore (3.2) and (3.3) are left out in this formulation.

Cost Function

The cost function is also modified. The deadhead cost between trips $s, t \in \mathcal{T}_i$ are the same as in the (SMILP). Between trips $s \in \mathcal{T}_i, t \in \mathcal{T}_j$ with $i < j$, there is an end point $t' \in \hat{\mathcal{P}}_i$ with t representing t' if t starts another partial duty in S_j. The deadhead cost $c^{\mathrm{d}}_{s,t}$ is treated in I_i as $c^{\mathrm{d}}_{s,t'}$ and the trip cost c^{t}_t is treated in I_j. Therefore the trip cost for the trip starting a duty arises additionally.

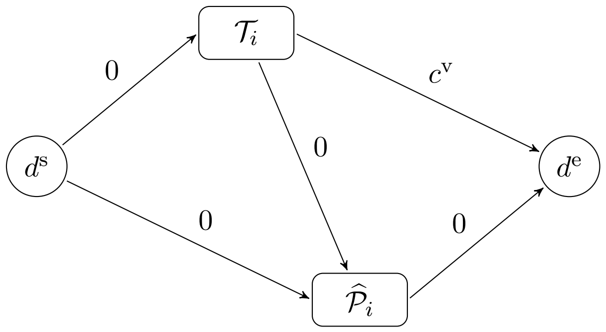

Figure 4.2: Graph $\overline{G}_i$ and vehicle cost for $i \in [n]\backslash\{1\}$

The fixed vehicle costs require a different treatment. If a duty ends with an end point, the vehicle cost of this duty arises already in the partial instance where the end point is created. Therefore, we use the vehicle cost only for duties that end with a trip. Thus, the arcs $\mathcal{T}_i \times d^{\mathrm{e}}$ have vehicle cost c^{v}. All other arcs that are incident with d^{s} or d^{e} have no vehicle cost. Figure 4.2 shows a simplified visualization of the partial task graph and the vehicle cost.

Besides the cover constraints, also the route costs are not treated here. They are specified in the heuristic. The formulation of the partial instance is called (SMILP$_i$) for $i \in [n]\backslash\{1\}$.

Solving Partial Instance I_1

As mentioned before, the partial instance I_1 plays a special role since the vehicles are introduced there. The vehicle set $\mathcal{V}$ is added to the partial task graph and all duties start with a vehicle. We show how the formulation (SMILP$_1$) differs from (SMILP$_i$) for $i \in [n]\backslash\{1\}$.

$$\begin{aligned}
\min \quad & \sum_{s\in\mathcal{T}_i} x_{s,d^{\mathrm{e}}} c^{\mathrm{v}} + \sum_{t\in\mathcal{T}_i} x_{d^{\mathrm{s}},t} c^{\mathrm{t}}_t \\
& \sum_{t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i} \sum_{s\in \mathrm{N}^-_{\overline{G}_i}(t)\setminus\{d^{\mathrm{s}}\}} \left[x_{s,t}\left(c^{\mathrm{d}}_{s,t} + c^{\mathrm{t}}_t\right) + \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}\left(c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t} - c^{\mathrm{d}}_{s,t}\right)\right] && (\mathrm{SMILP}_i)\\
\text{s.t.} \quad & \sum_{t\in \mathrm{N}^-_{\overline{G}_i}(s)} x_{t,s} = \sum_{t\in \mathrm{N}^+_{\overline{G}_i}(s)} x_{s,t} && \text{for all } s\in \overline{V}_i\setminus\{d^{\mathrm{s}},d^{\mathrm{e}}\} && (4.16)\\
& \sum_{s\in \mathrm{N}^-_{\overline{G}_i}(t)} x_{s,t} = 1 && \text{for all } t\in\hat{\mathcal{P}}_i && (4.11)\\
& \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} \le x_{s,t} && \text{for all } t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i, s\in \mathrm{N}^-_{\overline{G}_i}(t)\setminus\{d^{\mathrm{s}}\} && (4.17)\\
& f^0_s \le e_s && \text{for all } s\in\hat{\mathcal{P}}_i && (4.12)\\
& 0 \le e_s - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{s,r} && \text{for all } t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i, s\in \mathrm{N}^-_{\overline{G}_i}(t)\setminus\{d^{\mathrm{s}}\} && (4.18)\\
& e_t \le 1 - f^{\mathrm{t}}_t - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{r,t} && \text{for all } t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i, s\in \mathrm{N}^-_{\overline{G}_i}(t)\setminus\{d^{\mathrm{s}}\} && (4.19)\\
& e_t \le e_s - x_{s,t}\left(f^{\mathrm{d}}_{s,t} + f^{\mathrm{t}}_t\right) - \sum_{r\in\mathcal{R}_{s,t}} z_{s,r,t}\left(f^{\mathrm{d}}_{s,r} + f^{\mathrm{t}}_r + f^{\mathrm{d}}_{r,t} - f^{\mathrm{d}}_{s,t}\right) + (1 - x_{s,t}) \\
& && \text{for all } t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i, s\in \mathrm{N}^-_{\overline{G}_i}(t)\setminus\{d^{\mathrm{s}}\} && (4.20)\\
& e_s + f^{\mathrm{t}}_s \le f^{\max}_s + (1 - x_{d^{\mathrm{s}},s})\cdot\left(1 + f^{\mathrm{t}}_s\right) && \text{for all } s\in\mathcal{T}_i && (4.13)\\
& f^{\min}_s \le e_s + (1 - x_{s,d^{\mathrm{e}}}) && \text{for all } s\in\mathcal{T}_i && (4.14)\\
& \sum_{t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i} x_{d^{\mathrm{s}},t} \le |\mathcal{V}| && && (4.15)\\
& x_{s,t}\in\{0,1\} && \text{for all } (s,t)\in\overline{A}_i && (4.21)\\
& z_{s,r,t}\in\{0,1\} && \text{for all } t\in\mathcal{T}_i\cup\hat{\mathcal{P}}_i, s\in \mathrm{N}^-_{\overline{G}_i}(t)\setminus\{d^{\mathrm{s}}\}, r\in\mathcal{R}_{s,t} && (4.22)\\
& e_s\in[0,1] && \text{for all } s\in\overline{V}_i\setminus\{d^{\mathrm{s}},d^{\mathrm{e}}\} && (4.23)
\end{aligned}$$

Definition 12 (Partial Transformed Task Graph)**.** Let $i = 1$. For a set of vehicles $\mathcal{V}$, a set of end points $\hat{\mathcal{P}}_1$ and the partial trip set $\mathcal{T}_1$, the partial transformed task graph for I_1 is the directed graph $\overline{G}_1 = \left(\overline{V}_1, \overline{A}_1\right)$ with vertex set

$$\overline{V}_1 := \{d^{\mathrm{s}}, d^{\mathrm{e}}\} \mathbin{\dot\cup} \mathcal{V} \mathbin{\dot\cup} \mathcal{T}_1 \mathbin{\dot\cup} \hat{\mathcal{P}}_1$$

and arc set

$$\overline{A}_1 := (\{d^{\mathrm{s}}\} \times \mathcal{V}) \mathbin{\dot\cup} \left\{(s,t) \in (\mathcal{V} \mathbin{\dot\cup} \mathcal{T}_i) \times \left(\mathcal{T}_i \mathbin{\dot\cup} \hat{\mathcal{P}}_i\right) \mid s \prec t\right\} \mathbin{\dot\cup} \left(\left(\mathcal{V} \mathbin{\dot\cup} \mathcal{T}_i \mathbin{\dot\cup} \hat{\mathcal{P}}_i\right) \times \{d^{\mathrm{e}}\}\right).$$

With introducing the vehicles we have to ensure that each vehicle is considered exactly once and respect its initial fuel state. Hence we add the constraints

$$\sum_{s \in \mathrm{N}^-_{\overline{G}_1}(t)} x_{s,t} = 1 \qquad \text{for all } s \in \mathcal{V} \tag{3.5}$$

$$e_s \le f_s^0 \qquad \text{for all } s \in \mathcal{V} \tag{3.7}$$

to (SMILP$_1$).

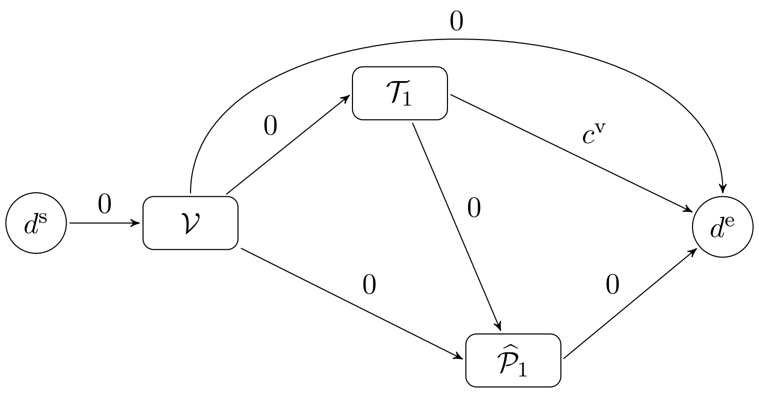

Figure 4.3: Graph $\overline{G}_1$ and vehicle cost for (SMILP$_1$)

The objective function is not modified. The vehicle cost is charged for each non-empty duty that does not end with an end point. The deadhead cost $c^{\mathrm{d}}_{s,t}$ or $c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t}$ for $s \in \mathcal{V}$ is respected in the objective function. Figure 4.3 shows a simplified visualization of the partial task graph $\overline{G}_1$ and the vehicle cost.

Model Equivalence

We examine whether the heuristic that we have developed in Section 4.1.2 and Section 4.1.3 provides feasible solutions to the problem formulation (SMILP). As mentioned before, this setting is only a basic framework for the actual heuristic where the cover constraints are not considered. Algorithm 1 iteratively creates partial solutions S_i for $i = \{n, \dots, 1\}$. Starting with an empty end point set $\hat{\mathcal{P}}_n = \emptyset$, the heuristic iteratively creates the end point set $\hat{\mathcal{P}}_i$ out of the previous partial solution S_{i+1}. The end point set $\hat{\mathcal{P}}_i$ is part of the formulation (SMILP$_i$).
We show that the partial solutions can be transformed to a feasible solution of the split formulation. Vice versa, a solution of the split formulation can be divided into feasible partial solutions. These results hold under the restrictions that the cover constraints are relaxed and the route costs are neglected.

Theorem 6 (Transformation of (SMILP$_i$) into (SMILP))**.** *Let $\{S_1, \dots, S_n\}$ be partial solutions, i. e. S_i is feasible in the (SMILP$_i$). For $i \in [n-1]$, the end point set $\hat{\mathcal{P}}_i$ has been created out of S_{i+1} according to Algorithm 1. Further holds $\hat{\mathcal{P}}_n = \emptyset$. The partial solutions can be transformed into a solution $\overline{S}$ that is feasible in the (SMILP) when relaxing the cover constraints (3.2) and (3.3), so that the values $\sum_{i=1}^{n} \operatorname{val}(S_i)$ and $\operatorname{val}(\overline{S})$ coincide.*

Proof. For $i \in [n]$, let $S_i = (x^i, z^i, e^i)$ be the partial solution of I_i. We construct a solution $\overline{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$. For this, we identify each end point set $\hat{\mathcal{P}}_i$ with the partial split point set $\mathcal{P}_i$. An end point $t \in \hat{\mathcal{P}}_i$ that represents a trip $s \in \bigcup_{j=i+1}^{n} \mathcal{T}_j$ is identified with the split point $\mathrm{SP}_i(s)$. Each split point with which no end point is identified is not used in $\overline{S}$.
Let $i \in [n]$ be arbitrary. For $i = 1$, let $s \in \mathcal{V} \cup \mathcal{T}_1$ and $t \in \mathcal{T}_1 \cup \mathcal{P}_1$. For $i > 1$, let $s \in \mathcal{T}_i$ and $t \in \mathcal{T}_i \cup \mathcal{P}_i$. We state

$$\bar{x}_{s,t} := \begin{cases} x^i_{s,t} & \text{if } t \in \mathcal{T}_i \cup \hat{\mathcal{P}}_i \\ 0 & \text{if } t \in \mathcal{P}_i \backslash \hat{\mathcal{P}}_i \end{cases} \qquad \bar{z}_{s,r,t} := \begin{cases} z^i_{s,r,t} & \text{if } t \in \mathcal{T}_i \cup \hat{\mathcal{P}}_i \\ 0 & \text{if } t \in \mathcal{P}_i \backslash \hat{\mathcal{P}}_i \end{cases} \qquad \text{for } r \in \mathcal{R}_{s,t}.$$

The term $t \in \mathcal{P}_i \backslash \hat{\mathcal{P}}_i$ means that Algorithm 1 has not created the respective end point and therefore the split point is not used in this solution.
Let $t \in \mathcal{T}_i$ and $s = \mathrm{SP}_j(t)$ with $j < i$. Then we state:

$$\bar{x}_{s,t} := \begin{cases} 1 & \text{if } s \in \hat{\mathcal{P}}_j \text{ and } x^j_{d^s,s} = 0 \\ 0 & \text{otherwise} \end{cases}$$

Consider $s \in \hat{\mathcal{P}}_j$ and $x^j_{d^s,s} = 1$. Then the end point is not appended to a partial duty. In this case, Algorithm 1 creates a new end point in $\hat{\mathcal{P}}_{j-1}$. If $x^j_{d^s,s} = 0$, then we have

$x^j_{t',s} = 1$ for some $t \in \mathcal{T}_j$ and therefore s is appended to a duty and no further end point is created.
Finally, we state

$$\begin{aligned}
\bar{x}_{d^{\mathrm{s}},v} &:= 1 && \text{for } v \in \mathcal{V} & \bar{x}_{s,d^{\mathrm{e}}} &:= x^i_{s,d^{\mathrm{e}}} && \text{for } s \in \mathcal{T}_i\\
\bar{e}_s &:= e^i_s && \text{for } s \in \mathcal{T}_i \cup \hat{\mathcal{P}}_i & \bar{e}_v &:= e^1_v && \text{for } v \in \mathcal{V}\\
\bar{e}_s &:= 0 && \text{for } s \in \mathcal{P} \setminus \bigcup_{i=1}^{n} \hat{\mathcal{P}}_i & \bar{u}_m &:= 0 && m \in \mathcal{M}.
\end{aligned}$$

We show that $\overline{S}$ is a feasible solution of the (SMILP). The flow conservation (4.2) is obviously fulfilled for $t \in \mathcal{V} \cup \mathcal{T} \cup \mathcal{P} \setminus \bigcup_{i=1}^n \hat{\mathcal{P}}_i$. Let $t \in \hat{\mathcal{P}}_i$ with $t = \mathrm{SP}_i(t')$ for some $t' \in \bigcup_{j=i+1}^n \mathcal{T}_j$. We have $\sum_{s \in \mathrm{N}^-_{G_i}(t)} x^i_{s,t} = 1$ due to (4.11) and $\mathrm{N}^+_G(t) = \{t'\}$. Thus

$$x^j_{d^{\mathrm{s}},t} + \sum_{s \in \mathrm{N}^-_G(t)} x_{s,t} = 1 \quad \text{and} \quad x^j_{d^{\mathrm{s}},t} + x_{t,t'} = 1 \quad \Rightarrow \quad \sum_{s \in \mathrm{N}^-_G(t)} x_{s,t} = \sum_{s \in \mathrm{N}^+_G(t)} x_{t,s}.$$

The fuel constraints hold in each partial solution individually, the initial fuel states f^0_t for $t \in \bigcup_{i=1}^n \mathcal{T}_i$ maintain feasible fuel states at the transitions between the partial instances. Therefore, $\overline{S}$ is feasible in the (SMILP) without the cover constraints (3.2) and (3.3).
In each partial instance, we distinguish the non-empty duties concerning the vehicle cost. For each partial duty that ends with a trip, the vehicle cost is charged. If it ends with an end point, no vehicle cost is charged, since the partial duty is prepended to an already existing duty. Therefore the vehicle costs are charged exactly once for each duty of $\overline{S}$. Each trip cost c^{t}_t and deadhead cost $c^{\mathrm{d}}_{s,t}$ or $c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t}$ occurs in exactly one partial instance. Because of $\bar{u}_m = 0$, the route cost is not considered. Therefore

$$\sum_{i=1}^n \mathrm{val}(S_i) = \mathrm{val}\left(\overline{S}\right).$$

□

Theorem 7 (Transformation of (SMILP) into (SMILP$_i$))**.** *Let $\overline{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ be a feasible solution of the (SMILP). Then $\overline{S}$ can be transformed into partial solutions $\{S_1, \ldots, S_n\}$ where S_i is feasible in (SMILP$_i$) for $i \in [n]$ and the values $\mathrm{val}(\overline{S})$ and $\sum_{i=1}^n \mathrm{val}(S_i)$ coincide when neglecting the route cost.*

Proof. Let $\overline{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ be a feasible solution of the (SMILP). We construct solutions $S_i = (x^i, z^i, e^i)$ for $i \in [n]$. Similarly to the proof of Theorem 6, we create an end point for each used split point. Let $t \in \mathcal{T}_i$ with $\bar{x}_{\mathrm{SP}_{i'}(t),t} = 1$ for some $i', i \in [n]$. Then,

for each $j \in \{i', \dots, i-1\}$ we create an end point $\mathrm{SP}_j(t)$ in $\hat{\mathcal{P}}_j$. We identify each end point with the split point from which it is created.
Let $i \in [n]$ be arbitrary. For $i = 1$, let $s \in \mathcal{V} \cup \mathcal{T}_1$ and $t \in \mathcal{T}_1 \cup \hat{\mathcal{P}}_1$. For $i > 1$, let $s \in \mathcal{T}_i$ and $t \in \mathcal{T}_i \cup \hat{\mathcal{P}}_i$. We state

$$x^i_{s,t} := \bar{x}_{s,t} \qquad z^i_{s,r,t} := \bar{z}_{s,r,t} \qquad \text{for } r \in \mathcal{R}_{s,t}.$$

Let $s \in \mathcal{T}_i$. Then we state

$$x^i_{d^{\mathrm{s}},s} := \max_{j\in[i-1]} \left(\bar{x}_{\mathrm{SP}_j(s),s}\right) \qquad x^i_{s,d^{\mathrm{e}}} := \bar{x}_{s,d^{\mathrm{e}}}.$$

For $s \in \hat{\mathcal{P}}_i$ with $s = \mathrm{SP}_i(t)$ for some $t \in \bigcup_{j=i+1}^n \mathcal{T}_j$, we state

$$x^i_{d^{\mathrm{s}},s} := \max_{j\in[i-1]} \left(\bar{x}_{\mathrm{SP}_j(t),t}\right) \qquad x^i_{s,d^{\mathrm{e}}} := 1.$$

We have $x^i_{d^{\mathrm{s}},s} = 1$, iff s is the start of a partial duty. This means for $s \in \mathcal{T}_i$ that s is connected to one of its split points, i. e. $\bar{x}_{\mathrm{SP}_j(s),s} = 1$ for some $j \in [i-1]$. For $s \in \hat{\mathcal{P}}_i$, Algorithm 1 creates a new end point out of s and one of these split points is connected to the original trip. We have $x^i_{s,d^{\mathrm{e}}} = 1$, iff s is the end of a partial duty. If s is a trip then s is the end of the whole duty, while each end point is the end of a partial duty. We further state

$$x^1_{d^{\mathrm{s}},v} := 1 \qquad x^1_{v,d^{\mathrm{e}}} := \bar{x}_{v,d^{\mathrm{e}}} \qquad e^1_v := \bar{e}_v \qquad \text{for } v \in \mathcal{V} \qquad e^i_s := \bar{e}_s \qquad \text{for } s \in \mathcal{T}_i \cup \hat{\mathcal{P}}_i.$$

Each S_i is a feasible solution of the (SMILP$_i$) and the objective values $\mathrm{val}(\overline{S})$ and $\sum_{i=1}^n \mathrm{val}\,(S_i)$ coincide up to the route cost. The proof works analogously to the proof of Theorem 6. The route cost is not considered in the objective function of (SMILP$_i$). □

4.2 Customer-dependent Splitting

In this section, we introduce the customer-dependent splitting. In contrast to the splitting performed by [Kno16], the trips are not split according to their start times but according to their customers' start times. Therefore all trips of a route and all routes of a customer are in the same partial trip set. The advantage is that the cover constraints can be applied easily in the respective subproblems. The problem is that this formulation is not equivalent to the original problem, i. e. duties that are feasible in (MMILP) can be cut off in this formulation. We show restrictions, in which the application of this splitting is sensible, though.

4.2.1 Basic Idea

At first, we define the customer-dependent formally and show how we adapt the (SMILP$_i$) such that the heuristic solution fulfills the cover constraints.

Splitting

In the customer-dependent splitting, all trips of the same customer are in the same partial trip set. Therefore, we define the splitting as follows:

Definition 13 (Customer-dependent splitting)**.** Given points in time c_i, $i \in [n-1]$ with $c_i < c_{i+1}$ for $i \in [n-2]$. We first define a splitting of the customers $\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{C}_i$ as

$$\mathcal{C}_i := \begin{cases} \{c \in \mathcal{C} \mid z_c^{\text{start}} \leq c_1\} & \text{for } i = 1 \\ \{c \in \mathcal{C} \mid c_{i-1} < z_c^{\text{start}} \leq c_i\} & \text{for } i \in [n-1] \backslash \{1\} \\ \{c \in \mathcal{C} \mid c_{n-1} < z_c^{\text{start}}\} & \text{for } i = n. \end{cases}$$

Based on the customer splitting, we define the splitting of $\mathcal{T}$ as

$$\mathcal{T}_i := \{t \in \mathcal{T} \mid (M \circ C)(t) \in \mathcal{C}_i\} \qquad \text{for } i \in [n].$$

Solving the Partial Instances

The formulation of the partial instances is built on the basic structure (SMILP$_i$). The cover constraints have not been considered there. We therefore introduce the decision variable $u_m \in \{0,1\}$ for $m \in C^{-1}(\mathcal{C}_i)$. Since a customer $c \in \mathcal{C}_i$ has all his trips in $\mathcal{T}_i$, only the cover constraints concerning these customers are included in partial instance I_i. We therefore add the following constraints:

$$\sum_{m \in C^{-1}(c)} u_m = 1 \qquad \text{for all } c \in \mathcal{C}_i \tag{4.24}$$

$$\sum_{s \in \mathrm{N}_{G_i}^{-}(t)} x_{s,t} = u_m \qquad \text{for all } m \in C^{-1}(\mathcal{C}_i), t \in m \tag{4.25}$$

$$u_m \in \{0,1\} \qquad \text{for all } m \in C^{-1}(\mathcal{C}_i) \tag{4.26}$$

Further the route costs are not considered in (SMILP$_i$) so far. We again have to consider only the route costs belonging to $c \in \mathcal{C}_i$. We therefore add the following term to the objective function:

$$\sum_{m \in C^{-1}(\mathcal{C}_i)} u_m c_m^{\mathrm{r}}$$

We call this formulation (CMILP$_i$) for $i \in [n]$. Let S be a solution that is created with Algorithm 1 and the respective partial solutions S_i are feasible in (CMILP$_i$) for $i \in [n]$. Then we call S a feasible solution of the (CMILP).

Model Equivalence

In the following, we examine whether the formulations (MMILP) and (CMILP) are equivalent. We show that each solution that is computed with the heuristic is actually a feasible solution of the original problem. On the other hand, we provide a counterexample, in which the optimal solution of the (MMILP) is not a feasible outcome of the heuristic.

Theorem 8 (Transformation of (CMILP) into (MMILP))**.** *Let S be a feasible solution of the* (CMILP)*. Then S is feasible in the* (MMILP) *and*

$$\text{cost}\,(S) = \sum_{i=1}^{n} \text{cost}\,(S_i)$$

Proof. Let S be a solution that is created with Algorithm 1 and the respective partial solutions S_i are feasible in (CMILP$_i$) for $i \in [n]$. The (CMILP$_i$) builds on the (SMILP$_i$) and additionally contains the variables u_m and the constraints (4.24), (4.25) and (4.26). Therefore each feasible solution of the (CMILP$_i$) is also feasible in (SMILP$_i$).
According to Theorem 6, solution S is feasible in (MMILP) except for the cover constraints. Let $c \in \mathcal{C}$ arbitrary. Then there is a unique $i \in [n]$ with $c \in \mathcal{C}_i$. In (CMILP$_i$) exist decision variables u_m for all $m \in C^{-1}(c)$ and then (3.2) follows directly from (4.24). Let $m \in \mathcal{M}$ arbitrary. There is a unique $i \in [n]$ with $m \in C^{-1}(\mathcal{C}_i)$ and all trips $t \in m$ are in (CMILP$_i$). Then (3.3) and (3.14) follow directly from (4.25) and (4.26). In the (CMILP$_i$), the route cost is added to the objective function. For each route, the route cost arises in exactly one partial instance. Therefore S is a feasible solution of the (MMILP) and $\text{cost}(S) = \sum_{i=1}^{n} \text{cost}\,(S_i)$. ☐

Theorem 8 shows that each heuristical solution is a feasible solution of the (MMILP). Now we show that a feasible solution of the (MMILP) is not necessarily feasible in the heuristic using a customer-dependent splitting.

Example 3. Let $\mathcal{T} = \{t_1, t_2, t_3\}$ with $t_1 \prec t_2 \prec t_3$ and the properties as shown in Table 4.1

Trip	Start	End	Route	Customer
t_1	8:00	8:15	m_1	b_1
t_2	8:30	8:45	m_2	b_2
t_3	9:00	9:15	m_1	b_1

Table 4.1: Trips corresponding to Example 3

We can see easily that the duty $d = (t_1, t_2, t_3)$ is a feasible solution.

If we now set a time point $c_1 := 8{:}15$, then the partial trip sets are $\mathcal{T}_1 = \{t_1, t_3\}$ and $\mathcal{T} = \{t_2\}$. There is one split point $\mathrm{SP}_1(t_2)$ with $z^{\text{start}}_{\mathrm{SP}_1(t_2)} = 8{:}30$ and thus $t_3 \not\prec \mathrm{SP}_1(t_2)$. The connection of t_2 and t_3 is not feasible in this heuristic and therefore the duty d is not feasible.

4.2.2 Quality of the Heuristic

Although the optimal solution is possibly not feasible in the heuristic, we examine the quality of feasible heuristical solutions. Based on an arbitrary feasible solution of the (MMILP), we inspect at which points this solution becomes infeasible in the heuristic and if we can construct a new solution which is feasible there. Depending on the initial solution, we aim to receive upper bounds for the total cost of the new solution.

Quality of Feasible Solutions

Let $d(v)$ be a duty of a feasible schedule according to Definition 5. The only reason that makes $d(v)$ infeasible in the customer-dependent heuristic is the following: A trip $s \in \mathcal{T}$ is positioned earlier in time than $t \in \mathcal{T}$, but is located in a later partial trip set, i. e. $s \prec t$ and $s \in \mathcal{T}_i, t \in \mathcal{T}_j$ with $j < i$. This may occur if

$$z^{\text{start}}_s < z^{\text{start}}_t \qquad \text{and} \qquad z^{\text{start}}_{(C\circ M)(s)} > z^{\text{start}}_{(C\circ M)(t)}.$$

Then the feasible connection between s and t is infeasible in the heuristic. Under certain conditions it is possible to create new duties d_1 and d_2 covering all trips of the duty $d(v)$ where d_1 and d_2 are both feasible in the heuristic. To realize these duties, we also need an additional vehicle that covers duty d_2. Having these additional vehicles, we construct a new solution whose total cost is bounded by twice the original cost.

Definition 14 (Customer extension and splitting length)**.** Consider a customer set $\mathcal{C}$ and time points c_i, $i \in [n-1]$ with $c_i < c_{i+1}$ for all $i \in [n-2]$. We define the following values:

- Customer Extension: $L_{\mathrm{C}} := \max\limits_{c\in\mathcal{C}} \left(z^{\text{end}}_c - z^{\text{start}}_c\right)$
- Splitting Length: $L_{\mathrm{S}} := \min\limits_{i\in[n-1]} \left(c_{i+1} - c_i\right)$

According to Definition 1, z^{start}_c is the earliest and z^{end}_c the latest start time of a trip corresponding to customer $c \in \mathcal{C}$. The value L_{C} describes the greatest time range between the earliest and the latest trip of any customer. The value L_{S} indicates the smallest time range between two splitting points.
If the customer extension is bounded by the splitting length, i. e. $L_{\mathrm{C}} \le L_{\mathrm{S}}$, the following advantage occurs: Let $t \in \mathcal{T}_i$ and $s \in \mathcal{T}$ with $s \prec t$. Then s is located not more

than one partial instance later than $\mathcal{T}_i$, i. e. $s \in \bigcup_{j=1}^{i+1} \mathcal{T}_j$. With this condition, we restrict the number of possible connections that are destroyed by the heuristic.
In the following lemma, we construct two heuristic-feasible duties d_1 and d_2 out of each duty d. This construction requires the customer extension to be bounded by the splitting length. It further requires an additional vehicle $v' \in \mathcal{V}$ that is assigned to the additional duty. Only in this lemma, we denote the vehicle duties as lists of trips and assume the respective refuel points to be given as additional information. We write $s \prec t$ if (s, t) is feasible in the (MMILP), i. e. $(s, t) \in A$. We write $s \rightarrow t$ if the connection (s, t) is feasible in (CMILP), i. e. $(s, t) \in \overline{A}$ or there is a split point $t' \in \mathcal{P}$ with $(s, t'), (t', t) \in \overline{A}$.

Lemma 3. *Let $S = (x, z, e, u)$ be a feasible solution of the (MMILP) and let c_i be time points for $i \in [n-1]$ with $c_i < c_{i+1}$ for $i \in [n-2]$. Let*

$$L_{\mathrm{C}} \leq L_{\mathrm{S}}. \tag{4.27}$$

Let $d = (v, t_1, \ldots, t_k)$ be a duty of S with vehicle $v \in \mathcal{V}$. If d is not feasible in (CMILP), let a be the smallest index such that $t_a \not\rightarrow t_{a+1}$ and let $v' \in \mathcal{V}$ with

$$z_{v'} + t_{v',t_{a+1}} \leq z^{\mathrm{start}}_{t_{a+1}} \tag{4.28}$$

and

$$f^0_{v'} \geq e_{t_{a+1}} + f^{\mathrm{d}}_{v',t_{a+1}} + f^{\mathrm{t}}_{t_{a+1}}. \tag{4.29}$$

Then there are duties d_1, d_2 with $d_1 \uplus d_2 = d \uplus \{v'\}$ such that d_1, d_2 are part of a feasible solution of the (CMILP).
Let additionally

$$c^{\mathrm{d}}_{v',t_{a+1}} \leq c^{\mathrm{d}}_{v,t_1} + \sum_{j=1}^{a} \left(c^{\mathrm{t}}_{t_j} + c^{\mathrm{d}}_{t_j,t_{j+1}} \right). \tag{4.30}$$

Then the cost of the duties d_1, d_2 is at most twice the original cost, i. e.

$$\mathrm{cost}\,(d_1) + \mathrm{cost}\,(d_2) \leq 2 \cdot \mathrm{cost}\,(d)\,.$$

We can imagine the underlying conditions as follows: The additional vehicle $v' \in \mathcal{V}$ starts its duty with the trip t_{a+1}. Condition (4.28) ensures that v' is at $p^{\mathrm{start}}_{t_{a+1}}$ at the desired time and (4.29) ensures that v' has a sufficient fuel level to fulfill the duty. (4.30) ensures that the deadhead cost $c^{\mathrm{d}}_{v',t_{a+1}}$ is smaller than the cost for driving from v to t_{a+1} in the original duty $d(v)$.

Proof. If $d(v) = (t_1, \dots, t_k)$ is feasible in CMILP, the result is obvious. Else, there exist subsequent trips whose connection is not feasible in the heuristic. Let a be the smallest index with $t_a \not\to t_{a+1}$.
Consider $s \prec t$ with $s \not\to t$ and customers $b_s := (M \circ C)(s)$ and $b_t := (M \circ C)(t)$. If s and t is in the same partial trip set, then $s \to t$. If s is in an earlier partial trip set, s and t are connected via a split point. Thus s is in a later partial trip set. Using $c_0 := -\infty, c_n := +\infty$, there are split points c_{l-1}, c_l, c_{l+1} for $l \in [n-1]$ with

$$z_s^{\text{start}} < z_t^{\text{start}} \qquad z_{b_t}^{\text{start}} \le c_l < z_{b_s}^{\text{start}} \qquad c_l + L_{\text{S}} \le c_{l+1} \qquad z_{b_s}^{\text{start}} \le z_s^{\text{start}} \le z_{b_s}^{\text{start}} + L_{\text{C}}.$$

From (4.27) follows:

$$\begin{aligned} z_{b_s}^{\text{start}} \le z_s^{\text{start}} < z_t^{\text{start}} \le z_{b_t}^{\text{start}} + L_{\text{C}} \le c_l + L_{\text{C}} \le c_l + L_{\text{S}} \le c_{l+1} \\ z_{b_t}^{\text{start}} \ge z_t^{\text{start}} - L_{\text{C}} > z_s^{\text{start}} - L_{\text{C}} \ge z_{b_s}^{\text{start}} - L_{\text{C}} > c_l - L_{\text{C}} \ge c_l - L_{\text{S}} \ge c_{l-1} \end{aligned}$$

and therefore $t \in \mathcal{T}_l, s \in \mathcal{T}_{l+1}$.
In summary, we have shown the following equivalence: Let $s \prec t$ and $t \in \mathcal{T}_l$. Then holds

$$s \to t \Leftrightarrow s \in \bigcup_{j=1}^{l} \mathcal{T}_j \qquad \text{and} \qquad s \not\to t \Leftrightarrow s \in \mathcal{T}_{l+1}.$$

Note that the relation $\preceq$ is an equivalence relation on d. Due to the cover constraints, there are no $s, t \in d$ with $(M \circ C)(s) = (M \circ C)(t)$ and $M(s) \ne M(t)$ and hence transitivity holds.

Time Feasibility

For all $i \in [k-2]$ holds: $t_i \prec t_{i+1} \prec t_{i+2}$ and therefore $t_i \prec t_{i+2}$. We show how we construct feasible duties d_1, d_2 such that $s \to t$ holds for all subsequent trips s and t in one duty. We initially set the duties

$$d_1 := (v, t_1, \dots, t_a) \qquad\qquad d_2 := (v', t_{a+1})$$

which is feasible by assumption. For each $i \in \{a+2, \dots, k\}$, let t be the current last trip of d_1. Append t_i to d_1 if $t \to t_i$, else to d_2. We prove that t_i can always be feasibly appended to one of the duties:
In each step, choose $l_1, l_2 \in [n]$ as the indices such that for the current last trip t of d_1 and d_2 holds $t \in \mathcal{T}_{l_1}$ and $t \in \mathcal{T}_{l_2}$, respectively. We prove the feasibility by induction over $i \in \{a+1, \dots, k\}$.

- Induction Base: Trip t_{a+1} is feasibly appended to d_2 because of (4.28). After appending t_{a+1} holds $l_1 > l_2$ because of $t_a \not\to t_{a+1}$.

- Induction Hypothesis: For each $i \in \{a+2, \dots, k\}$, trip t_i can be appended to d_1 or d_2. Afterwards still holds $l_1 > l_2$.
- Induction Step: Choose $l \in [n]$ such that $t_i \in \mathcal{T}_l$. Since $t_j \prec t_i$ for all $j \in [i-1]$ holds $l+1 \geq l_1$ and since $l_1 > l_2$ holds $l \geq l_2$. Thus t_i can be appended to d_1 or d_2. If $l < l_1$ then t_i is appended to d_2 and $l_2 := l < l_1$. Else t_i is appended to d_1 and $l_1 := l > l_2$. Therefore still holds $l_1 > l_2$.

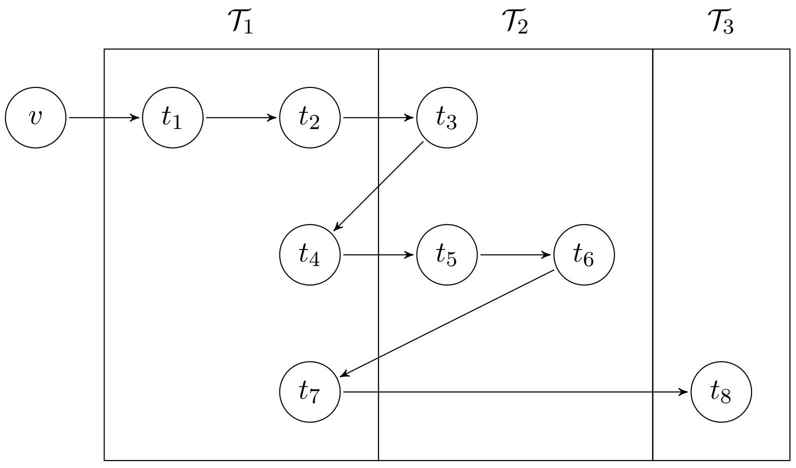

Figure 4.4: Original duty $d(v)$

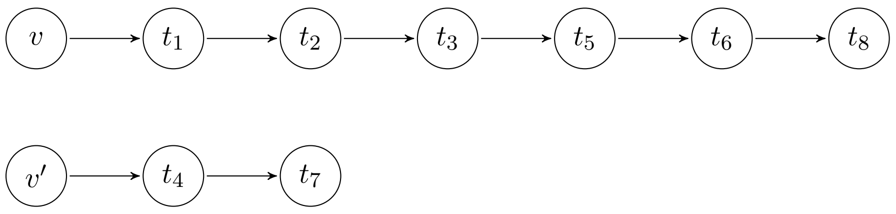

Figure 4.5: Constructed duties $d_1\,(v)$ and $d_2\,(v')$

Figure 4.4 provides an example for a duty $d(v)$. The boxes indicate in which partial trip set the trips are located. Figure 4.5 shows how the duties $d_1(v)$ and $s_2\,(v')$ are constructed. In this example, it is crucial that t_5 is appended to t_3 although a connection to t_4 would also be feasible. If t_5 were appended to t_4 then there is no available duty for t_7.

In summary, we have proven that each trip $t \in d$ can be feasibly appended either to d_1 or to d_2 and the vehicles v and v' are feasibly assigned to the duties d_1 and d_2, respectively.

Fuel Feasibility

After proving that we can construct duties d_1 and d_2 that are feasible in (SMILP) w. r. t. time, we examine the fuel states in the duties and visiting the refuel points. We consider d_1, d_2 as constructed in the previous part. Let $i \in [n-1]$ and let $z_{t_i,r,t_{i+1}} = 1$ for some $r \in \mathcal{R}_{t_i,t_{i+1}}$, i. e. refuel point r is visited between the trips t_i and t_{i+1}. We distinguish the following cases:

- $i < a$: Then $t_i, t_{i+1} \in d_1$ and r is set between t_i and t_{i+1} in d_1.
- $i \geq a$: Refuel point r is inserted in both duties. Let $j_1^- := \max\{j \leq i \mid t_j \in d_1\}$, $j_1^+ := \min\{j > i \mid t_j \in d_1\}$, j_2^-, j_2^+ analogously for d_2. If j_1^+ exists, insert r between $t_{j_1^-}$ and $t_{j_1^+}$ in duty d_1. If j_2^+ exists, insert r between $t_{j_2^-}$ and $t_{j_2^+}$ in duty d_2.

We now regard duty d_1 and simply write $t^- := t_{j_1^-}$ and $t^+ := t_{j_1^+}$. We prove that there is a refuel point copy $r' \in \mathcal{R}_{t^-,t^+}$ that belongs to the same refuel point as $r \in \mathcal{R}_{t_i,t_{i+1}}$. For simplicity of notation, we define the trip time $t_t := z_t^{\text{end}} - z_t^{\text{start}}$ for $t \in \mathcal{T}$.
From (2.6) and $r \in \mathcal{R}_{t_i,t_{i+1}}$ follows

$$\begin{aligned}
&z_{t^-}^{\text{end}} + t_{t^-,r'} + t_{r',t^+}\\
\leq &z_{t^-}^{\text{end}} + \sum_{j=j_1^-+1}^{i} \left(t_{t_{j-1},t_j} + t_{t_j}\right) + t_{t_i,r} + t_{r,t_{i+1}} + \sum_{j=i+1}^{j_1^+-1} \left(t_{t_j} + t_{t_j,t_{j+1}}\right)\\
\leq &z_{t_i}^{\text{end}} + t_{t_i,r} + t_{r,t_{i+1}} + \left(z_{t^+}^{\text{start}} - z_{t_{j+1}}^{\text{start}}\right)\\
\leq &z_{t^+}^{\text{start}}
\end{aligned}$$

and therefore $r' \in \mathcal{R}_{t^-,t^+}$. From (2.6) we can also see that the refueling time between t^- and t^+

$$\left(z_{t^+}^{\text{start}} - t_{r,t^+}\right) - \left(z_{t^-}^{\text{end}} + t_{t^-,r}\right) \geq \left(z_{t_{i+1}}^{\text{start}} - t_{r,t_{i+1}}\right) - \left(z_{t_i}^{\text{end}} + t_{t_i,r}\right)$$

is longer than the refueling time between t_i and t_{i+1} and thus the negative fuel consumption is smaller. Therefore

$$f_{r'}^{\text{t}} \leq f_r^{\text{t}}. \tag{4.31}$$

This works analogously for d_2 with $t_{j_2^-}$ and $t_{j_2^+}$. We apply this procedure to all refuel points in the original duty d.
Let $e_t \in [0,1]$ be feasible fuel states for $t \in d$. We prove that these fuel states are still feasible in d_1 and d_2. For the first trips, we distinguish the duties:

- Duty d_1: The values of e_v and $e_{t_1}, \ldots, e_{t_a}$ are still feasible since f_v^0 and all the trip connections do not change.
- Duty d_2: From condition (4.29) follows directly that $e_{t_{a+1}}$ is a feasible fuel state.

Let $t_i, t_{i'}$ be subsequent trips in d_1 or d_2 with no refuel point in-between. From (2.7) and (3.10) follows that

$$e_{t_{i'}} \leq e_{t_i} - \sum_{j=i+1}^{i'} \left(f^{\mathrm{d}}_{t_{j-1},t_j} + f^{\mathrm{t}}_{t_j} \right) \leq e_{t_i} - \left(f^{\mathrm{d}}_{t_i,t_{i'}} + f^{\mathrm{t}}_{t_{i'}} \right).$$

From (4.31) we additionally see that the fuel states are still feasible if a refuel point r' lies between t_i and t_{i+1}.
So far, we have neglected the case that more than one refuel point is visited between two trips of the same duty. This is for example $d = (v, t_1, r_1, t_2, r_2, t_3)$ with $t_1, t_3 \in d_1$ and $t_2 \in d_2$. As claimed in Section 2.2, it is not allowed to visit more than one refuel point between two trips. Therefore we assume without proof that we can insert some refuel point $r \in \mathcal{R}$ between each pair of subsequent trips and receive a feasible solution.
In summary, we have proven that the refuel points can be visited in d_1 and d_2 analogously to the original duty and then the original fuel states are still feasible in the new duties.

Costs

After constructing the duties d_1 and d_2, we show that the cost of the duties is not more than twice the original cost.
We first prove that $\mathrm{cost}\,(d_1) \leq \mathrm{cost}\,(d)$. The vehicle cost c^{v} is the same in d_1 and d. The trip cost c^{t} of d_1 is smaller than the trip cost of d since $d_1 \subseteq d$. The deadhead cost c^{d}_{v,t_1} coincides since the same vehicle is used in d_1 and d. All other deadhead costs of d_1 are smaller due to (2.8).
For duty d_2, we first regard the deadhead cost $c^{\mathrm{d}}_{v',t_{a+1}}$. Due to condition (4.30), the total cost up to trip t_{a+1} is smaller than in the original duty. All subsequent trip and deadhead costs are smaller as argued for d_1. Also the vehicle cost c^{v} is the same in d_2 and d.
Therefore we have

$$\mathrm{cost}\,(d_1) + \mathrm{cost}\,(d_2) \leq 2 \cdot \mathrm{cost}\,(d)\,.$$

This concludes the proof. $\square$

Remark 5. Lemma 3 also holds if instead of (4.28), (4.29) and (4.30) there is a $r \in \mathcal{R}_{v',t_{a+1}}$ such that

$$z_{v'} + t_{v',r} + t_{r,t_{a+1}} \leq z^{\text{start}}_{t_{a+1}} \tag{4.32}$$

$$f^0_{v'} - \left(f^{\text{d}}_{v',r} + f^{\text{t}}_r + f^{\text{d}}_{r,t_{a+1}}\right) \geq e_{t_{a+1}} + f^{\text{t}}_{t_{a+1}} \tag{4.33}$$

$$c^{\text{d}}_{v',r} + c^{\text{d}}_{r,t_{a+1}} \leq c^{\text{d}}_{v,t_1} + \sum_{j=1}^{a} \left(c^{\text{t}}_j + c^{\text{d}}_{j,j+1}\right). \tag{4.34}$$

Then we initially have $d_2 := (v', r, t_{a+1})$ and the modified conditions ensure that d_2 is still feasible and $\text{cost}\,(d_2) \leq \text{cost}\,(d)$. These conditions give us more flexibility w. r. t. the initial fuel of v'.

Given a duty of a solution that is feasible in the (MMILP) and an additional vehicle that fulfills certain conditions, we can construct a second duty, such that the duties are feasible in the heuristic, contain the same trips and the cost of the duties doubles at most. Based on this result, we aim to construct a feasible solution of the (CMILP) with at most twice the original cost. For this, we particularly need an additional vehicle set, with which the additional duties are covered. We examine conditions of the additional vehicle set in order to make this procedure possible. Since we do not know the solution in advance, we cannot make assumptions based on this solution as we did in Lemma 3, but we need to generalize these conditions to all feasible solutions.

Definition 15 (Conditions)**.** Let $\mathcal{T}$ be a trip set and c_i time points for $i \in [n-1]$ with $c_i < c_{i+1}$ for $i \in [n-2]$. Let $\mathcal{V}$ and $\mathcal{V}'$ with $\mathcal{V} \cap \mathcal{V}' = \emptyset$ be vehicle sets.

1. $\mathcal{V}$ and $\mathcal{V}'$ fulfill the *Feasibility Condition* if each $v \in \mathcal{V}$ can be assigned to some $v' \in \mathcal{V}'$ such that for all $t \in \mathcal{T}$ with $\{s \in \mathcal{T} \mid v \prec s, s \prec t, s \not\to t\} \neq \emptyset$ and $z^{\text{start}}_t > c_1$ one of the following conditions is fulfilled:
   - Feasibility condition without refuel point:
   $$z_{v'} + t_{v',t} \leq z^{\text{start}}_t \tag{4.35}$$
   $$f^0_{v'} - f^{\text{d}}_{v',t} \geq f^0_v - \min\left\{f^{\text{d}}_{v,t}, \min_{r' \in \mathcal{R}_{v,t}} \left(f^{\text{d}}_{v,r'} + f^{\text{t}}_{r'} + f^{\text{d}}_{r',t}\right)\right\} \tag{4.36}$$
   - Feasibility condition with refuel point: There is some $r \in \mathcal{R}_{v',t}$ such that
   $$z_{v'} + t_{v',r} + t_{r,t} \leq z^{\text{start}}_t \tag{4.37}$$
   $$f^0_{v'} - \left(f^{\text{d}}_{v',r} + f^{\text{t}}_r + f^{\text{d}}_{r,t}\right) \geq f^0_v - \min\left\{f^{\text{d}}_{v,t}, \min_{r' \in \mathcal{R}_{v,t}} \left(f^{\text{d}}_{v,r'} + f^{\text{t}}_{r'} + f^{\text{d}}_{r',t}\right)\right\} \tag{4.38}$$
2. $\mathcal{V}$ and $\mathcal{V}'$ fulfill the *Cost Condition* if each $v \in \mathcal{V}$ can be assigned to some $v' \in \mathcal{V}'$ such that for all $t \in \mathcal{T}$ with $z^{\text{start}}_t > c_1$ and $\{s \in \mathcal{T} \mid v \prec s, s \prec t, s \not\to t\} \neq \emptyset$ one of the following conditions is fulfilled:

- Cost condition without refuel point: (4.35), (4.36) and for all $s \in \mathcal{T}$ with $v \prec s, s \prec t, s \not\to t$ holds:

$$c^{\mathrm{d}}_{v',t} \leq c^{\mathrm{d}}_{v,s} + c^{\mathrm{t}}_{s} + c^{\mathrm{d}}_{s,t} \tag{4.39}$$

- Cost condition with refuel point: There is some $r \in \mathcal{R}_{v',t}$ such that (4.37), (4.38) and for all $s \in \mathcal{T}$ with $v \prec s, s \prec t, s \not\to t$ holds:

$$c^{\mathrm{d}}_{v',r} + c^{\mathrm{d}}_{r,t} \leq c^{\mathrm{d}}_{v,s} + c^{\mathrm{t}}_{s} + c^{\mathrm{d}}_{s,t} \tag{4.40}$$

We need the Feasibility Condition and the Cost Condition to extend the result of Lemma 3 from a single duty to a complete solution. The conditions cover all possible duties in which the heuristic-dependent infeasibilities may arise. The Feasibility Condition ensures the following: If the vehicle v can cover a certain trip $t \in \mathcal{T}$, the additional vehicle v' can also cover this trip and its fuel state at the arrival at p^{start}_t is at least the fuel state of v. In the Feasibility Condition with refuel point, the vehicle v' may visit a refuel point in-between. This makes it easier to fulfill the fuel condition, but harder to fulfill the time condition. The Cost Condition additionally requires that the cost of v' driving directly to trip t is at most the cost of v driving to t in a heuristic-infeasible duty. Note that all these conditions are obviously fulfilled if $\mathcal{V}'$ is a copy of $\mathcal{V}$, i. e. the start position, start time and the initial fuel state coincide for each $v \in \mathcal{V}$ and the $v' \in \mathcal{V}'$ that is assigned to v.
We now construct a heuristic-feasible solution S' whose cost doubles at most compared to the original solution S.

Theorem 9. *Let S be a feasible solution of the* (MMILP) *which is computed using the vehicle set $\mathcal{V}$. Let c_i be time points for $i \in [n-1]$ with $c_i < c_{i+1}$ for $i \in [n-2]$ and let*

$$L_{\mathrm{C}} \leq L_{\mathrm{S}}. \tag{4.27}$$

Let $\mathcal{V}'$ with $\mathcal{V} \cap \mathcal{V}' = \emptyset$ be a vehicle set such that $\mathcal{V}$ and $\mathcal{V}'$ fulfill the Feasibility Condition according to Definition 15. Using the vehicle set $\mathcal{V}' \cup \mathcal{V}$, there exists a feasible solution S' of the (CMILP) *with*

$$\operatorname{duties}(S') \leq 2 \cdot \operatorname{duties}(S). \tag{4.41}$$

If $\mathcal{V}$ and $\mathcal{V}'$ additionally fulfill the Cost Condition according to Definition 15, there exists a feasible solution S' of the (CMILP) *with*

$$\operatorname{cost}(S') \leq 2 \cdot \operatorname{cost}(S). \tag{4.42}$$

Proof. Let $S = (x, z, e, u)$ be a feasible solution of the (MMILP). We prove that each duty d of S fulfills the conditions of Lemma 3 or Remark 5. Let $v \in \mathcal{V}$ be the vehicle that covers d and $v' \in \mathcal{V}' \setminus \mathcal{V}$ be the vehicle assigned to v.

If d is not feasible in (CMILP), let a be the smallest index with $t_a \not\rightarrow t_{a+1}$ in d. Then holds $t_a \notin \mathcal{T}_1$ and therefore

$$z^{\text{start}}_{t_{a+1}} > z^{\text{start}}_{t_a} \geq z^{\text{start}}_{(M \circ C)(t_a)} > c_1.$$

Further we have $v \prec t_a$, $t_a \prec t_{a+1}$ and $t_a \not\rightarrow t_{a+1}$ and therefore $z^{\text{start}}_{t_{a+1}} > c_1$ and $\{s \in \mathcal{T} \mid v \prec s, s \prec t_{a+1}, s \not\rightarrow t_{a+1}\} \neq \emptyset$. For v' and t_{a+1} hold either (4.35) and (4.36) or (4.37) and (4.38) for some $r \in \mathcal{R}_{v',t_{a+1}}$. The implications (4.35) $\Rightarrow$ (4.28) or (4.37) $\Rightarrow$ (4.32), respectively, can be seen easily.
We can estimate the fuel state $e_{t_{a+1}}$ from above by the maximal fuel of vehicle v after serving t_{a+1}, i. e.

$$e_{t_{a+1}} + f^{\text{t}}_{t_{a+1}} \leq f^0_v - \min \left\{ f^{\text{d}}_{v,t_{a+1}}, \min_{r' \in \mathcal{R}_{v,t_{a+1}}} \left(f^{\text{d}}_{v,r'} + f^{\text{t}}_{r'} + f^{\text{d}}_{r',t_{a+1}} \right) \right\}$$

and therefore (4.29) follows from (4.36) by

$$\begin{aligned} f^0_{v'} &\geq f^0_v - \min \left\{ f^{\text{d}}_{v,t_{a+1}}, \min_{r' \in \mathcal{R}_{v,t_{a+1}}} \left(f^{\text{d}}_{v,r'} + f^{\text{t}}_{r'} + f^{\text{d}}_{r',t_{a+1}} \right) \right\} + f^{\text{d}}_{v',t} \\ &\geq e_{t_{a+1}} + f^{\text{d}}_{v',t_{a+1}} + f^{\text{t}}_{t_{a+1}}. \end{aligned}$$

The implication (4.38) $\Rightarrow$ (4.33) works analogously.
Therefore we can apply Lemma 3 or Remark 5 for each duty of S individually. For each duty d^v, we receive duties d^v_1 and d^v_2, starting with vehicles v and v'. If d^v is already feasible, the duty d^v_2 is empty. The duties d^v_1 and d^v_2 are feasible in (CMILP). We construct the new solution S' with all new duties as constructed in Lemma 3. For this, we need the vehicles $v \in \mathcal{V}$ for the duties d^v_1 and the vehicles $v' \in \mathcal{V}'$ for the duties d^v_2. Since these duties are feasible in the heuristic, the time constraints and fuel constraints are not violated by S'. Since the set of covered trips has not changed during the process, that cover constraints are not violated, too. Therefore, the new solution S' is a feasible solution of the (CMILP) and

$$\text{duties}\,(S') \leq 2 \cdot \text{duties}\,(S)\,.$$

Assume that $\mathcal{V}$ and $\mathcal{V}'$ additionally fulfill the Cost Condition. As shown before, we have $t_a \in \{s \in \mathcal{T} \mid v \prec s, s \prec t_{a+1}, s \not\rightarrow t_{a+1}\}$ and therefore (4.30) follows from (2.8) and (4.39) by

$$\begin{aligned} c^{\text{d}}_{v',t_{a+1}} &\leq c^{\text{d}}_{v,t_a} + c^{\text{t}}_{t_a} + c^{\text{d}}_{t_a,t_{a+1}} \\ &\leq c^{\text{d}}_{v,t_1} + \sum_{j=1}^{a} \left(c^{\text{t}}_{t_j} + c^{\text{d}}_{t_j,t_{j+1}} \right). \end{aligned}$$

The implication (4.40) $\Rightarrow$ (4.34) works analogously.
The cost of solution S' comprises the vehicle cost c^{v}, the deadhead cost c^{d}, the fuel cost c^{t} and the route cost c^{r}. Except from the route cost, all costs are already contained in the duty cost. These costs are bounded by $\text{cost}\,(d_1) + \text{cost}\,(d_2) \leq 2 \cdot \text{cost}\,(d)$. As mentioned before, the set of covered trips has not changed and thus the route costs have not changed either. Therefore, we have

$$\text{cost}\,(S') \leq 2 \cdot \text{cost}\,(S)\,.$$

□

To sum these results up, we have shown that we can always find a solution that is feasible in the customer-dependent heuristic and its total cost is not more than twice the optimal solution. For this result, we need the condition $L_{\mathrm{C}} \leq L_{\mathrm{S}}$. This condition applies for realistic instances, as we will discuss later. Further we need an additional vehicle set with the same size as the original vehicle set and the vehicles have certain requirements concerning their start positions and fuel states. As we regard the problem from the car sharing supplier's point of view, he is able to provide additional vehicles with the demanded properties in order to satisfy the travel request of the customers.
It is important to keep in mind that Theorem 9 does not provide an approximation factor for the customer-dependent heuristic. We can neither guarantee that we receive a solution S with $\text{cost}\,(S) \leq 2 \cdot \text{cost}\,(S^*)$, if we apply Algorithm 1 and solve each subproblem with (CMILP$_i$) to optimality (not even with the conditions of Definition 15). Nor can we ensure that an existing solution S which is computed with the customer-dependent heuristic fulfills $\text{cost}\,(S) \leq 2 \cdot \text{cost}\,(S^*)$. In this context S^* is an optimal solution. Theorem 9 only says that there exists a solution S' where each subproblem is feasible in (CMILP$_i$) and the total cost is at most twice the optimal cost.

Approximation Factor

Finally, we show that the developed heuristic is not a constant-factor approximation. We provide an example where the ratio between the objective values of the heuristic solution and the optimal solution is arbitrarily large.

Example 4. Let $K \leq 0$ arbitrary. Let $\mathcal{V} = \{v\}$ and $\mathcal{T} = \{t_1, t_2, t_3\}$ with the properties of Table 4.2.
We set all deadhead cost equal to the time, i. e. $c^{\mathrm{d}}_{s,t} = t_{s,t}$. Then we increase all deadhead cost leading to t_3 by the parameter K. Therefore holds:

$$c^{\mathrm{d}}_{v,t_3} = K + 2 \qquad\qquad c^{\mathrm{d}}_{t_1,t_3} = K$$

Trip	z_t^{start}	z_t^{end}	$M(t)$	$(M \circ C)(t)$	c_t^{t}
t_1	1	2	m_1	b_1	1
t_2	3	5	m_2	b_2	2
t_3	3	4	m_3	b_2	1

(a) Trips

	t_1	t_2	t_3
v	1	3	3
t_1		1	1

(b) Time between trips

Table 4.2: Trip Data for Example 4

For sake of completeness, we define the additional values:

$$\begin{array}{lllll} \mathcal{R} = \emptyset & f_v^0 = 1 & f_t^{\text{t}} = 0.1 & f_{s,t}^{\text{d}} = 0.1 & \text{for all } s \in \mathcal{V} \cup \mathcal{T}, t \in \mathcal{T} \\ c^{\text{v}} = 0 & & & c_m^{\text{r}} = 0 & \text{for all } m \in \mathcal{M} \end{array}$$

The fuel constraints are chosen such that all solutions are feasible w. r. t. fuel. We have $v \prec t$ for all $t \in \mathcal{T}$ and $t_1 \prec t_2, t_1 \prec t_3$. For customer b_1 the trip t_1 is fulfilled, for customer b_2 either trip t_2 or t_3. The optimal solution S^* consists of one duty d^* with

$$d^* = (v, t_1, t_2) \qquad \text{cost}(S^*) = 5.$$

We solve this problem via Algorithm 1 with $n = 2$ and $c_1 = 2$. Then the partial trip sets are $\mathcal{T}_1 = \{t_1\}$ and $\mathcal{T}_2 = \{t_2, t_3\}$. We first solve partial instance I_2 where we chose the cheaper trip t_3 and receive cost $(S_2) = 1$. Therefore the heuristic solution S consists of duty d with

$$d = (v, t_1, t_3) \qquad \text{cost}(S) = K + 3.$$

Figure 4.6 illustrates this example. The solid line indicates the heuristical solution, the dashed line stands for the optimal solution.

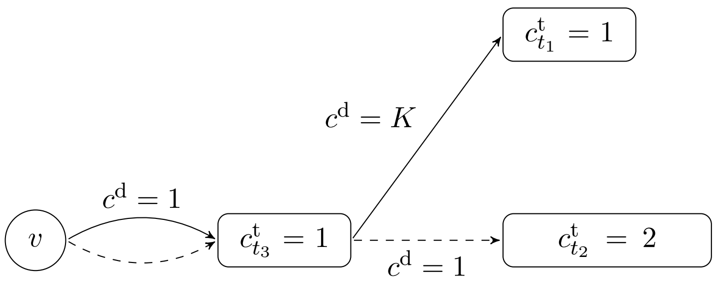

Figure 4.6: Illustration of Example 4

In summary, we have for a heuristic solution S and an optimal solution S^*

$$\frac{\text{cost}\,(S)}{\text{cost}\,(S^*)} \in \Theta(K)$$

in this example.

This is a remarkable result. In the original version of the Successive Heuristics without customers, the heuristic always computes a solution S with $\text{cost}\,(S) \leq n \cdot \text{cost}\,(S^*)$ (cf. [Kno16, p. 130]).

4.3 Time-dependent Splitting

The developed formulation (CMILP) based on a customer-dependent splitting is not equivalent to the original formulation (MMILP). This is shown in Example 3. Now, we aim to develop another heuristic such that the heuristic formulation is equivalent to the original formulation (MMILP), i. e. each feasible solution can be converted to a heuristic feasible solution. We develop a splitting on which the heuristic is based. To avoid a scenario as in Example 3, it is necessary that trips of the same customer may be in different partial trip sets. This leads to the following problem: If the partial instances are solved successively, we need a possibility to still guarantee the customer satisfaction for the entire problem. This has to be applied already in the first partial instance where a certain customer is considered, although we do not have any knowledge about the trips of the same customer in the partial instances to be solved later.

4.3.1 Basic Idea

We define a time-dependent splitting similar to [Kno16]. Based on this splitting, we adapt the model and describe the necessary cost estimation.

Splitting

We split the trip set $\mathcal{T}$ according to the start times of the trips.

Definition 16 (Time-dependent Splitting)**.** Given time points c_i for $i \in [n-1]$ with $c_i < c_{i+1}$ for $i \in [n-2]$. We define the splitting of $\mathcal{T}$ as

$$\mathcal{T}_i := \begin{cases} \{t \in \mathcal{T} \mid z_t^{\text{start}} \leq c_1\} & \text{for } i = 1 \\ \{t \in \mathcal{T} \mid c_{i-1} < z_t^{\text{start}} \leq c_i\} & \text{for } i \in [n-1] \backslash \{1\} \\ \{t \in \mathcal{T} \mid c_{n-1} < z_t^{\text{start}}\} & \text{for } i = n. \end{cases}$$

We denote the formulation (SMILP) with a splitting according to Definition 16 as (TMILP).

Solving the Partial Instances

Since the trips of the same customer may be in different partial trip sets, we cannot easily guarantee the customer satisfaction in just one partial instance. For each customer, we choose a route in the earliest solved partial instance where a trip of this customer arises. This route choice is definite, i. e. all trips of this customer in other partial instances are fulfilled if and only if their route is chosen. We illustrate this with a short example.

Example 5. Let $\mathcal{C} = \{b_1, b_2\}$, $\mathcal{M} = \{m_1, \ldots, m_4\}$ and $\mathcal{T} = \{t_1, \ldots, t_6\}$ with the properties as shown in Table 4.3.

Customer	Routes	Trips	
b_1	$\{m_1, m_2\}$	$m_1 = (t_1, t_5)$	$m_2 = (t_2, t_6)$
b_2	$\{m_3, m_4\}$	$m_3 = (t_3)$	$m_4 = (t_4)$

Table 4.3: Routes corresponding to Example 5

We consider a splitting with $\mathcal{T}_1 = \{t_1, \ldots, t_4\}$ and $\mathcal{T}_2 = \{t_5, t_6\}$. In the partial instance I_2, only customer b_1 is concerned and we choose a route for b_1. Thus, S_2 is a feasible partial solution if either t_5 or t_6 is fulfilled there. Assume that the route m_1 is chosen. In the partial instance I_1, the customers b_1 and b_2 are concerned. Since the route choice for b_1 is definite, we fulfill t_1, neglect t_2 and choose a route for b_2. Thus, S_1 is a feasible partial solution if t_1 is fulfilled, t_2 is not fulfilled and either t_3 or t_4 are fulfilled.

The partial instances $\{I_1, \ldots, I_n\}$ are solved in reversed order. Thus we define for a customer c the partial instance which is solved earliest and c is concerned as follows:

$$\gamma : \mathcal{C} \to [n] \qquad \gamma(c) := \max\left\{i \in [n] \mid \left((M \circ C)^{-1}(c) \cap \mathcal{T}_i\right) \neq \emptyset\right\}$$

Depending on γ and $\{\mathcal{T}_1, \ldots \mathcal{T}_n\}$ we define a partition $\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{C}_i$ as

$$\mathcal{C}_i := \{c \in \mathcal{C} \mid \gamma(c) = i\} \qquad \text{for } i \in [n].$$

Consider an arbitrary customer $c \in \mathcal{C}$. In the partial instance $I_{\gamma(c)}$, a multimodal route $m \in C^{-1}(c)$ is chosen and this choice is definite. This means, in all subsequently solved partial instances, all trips $t \in m$ are fixed to be fulfilled in advance and all trips $t \in (M \circ C)^{-1}(c) \setminus m$ are fixed to be neglected.
In the partial trip set $\mathcal{T}_{\gamma(c)}$ there is at least one trip of c. But there are also trips of c that are in other partial trip sets. There are even routes with no trip in $\mathcal{T}_{\gamma(c)}$ at all. These routes must not be neglected. Therefore, we need a method to choose the routes

where all routes $m \in C^{-1}(c)$ are considered. For this, we try to estimate the total cost of the routes in advance. Solving the partial instances is based on the (SMILP$_i$).
We consider a partial instance I_i. For the cover constraints, we introduce the decision variable $u_m \in \{0,1\}$ for $m \in C^{-1}(\mathcal{C}_i)$. In the customer constraints, only the customers in $\mathcal{C}_i$ are considered. The route constraints are restricted to the trips in $\mathcal{T}_i$. The cover constraints read as follows:

$$\sum_{m \in C^{-1}(c)} u_m = 1 \qquad \text{for all } c \in \mathcal{C}_i \tag{4.43}$$

$$\sum_{s \in \mathrm{N}^-_{\overline{G}_i}(t)} x_{s,t} = u_m \qquad \text{for all } m \in C^{-1}(\mathcal{C}_i), t \in m \cap \mathcal{T}_i \tag{4.44}$$

$$u_m \in \{0,1\} \qquad \text{for all } m \in C^{-1}(\mathcal{C}_i) \tag{4.45}$$

For the constraint (4.43) it is irrelevant, if the considered route has a trip in $\mathcal{T}_i$.
After solving the partial instance, all u_m are fixed for the partial instances that are solved later. The fixed route decisions from the previously solved partial instances have an impact on this instance, too.
We define terms with which we ensure the fixed route choices:

Definition 17. Let $i \in [n]$. We define

$$\overline{\mathcal{C}}_i := \{c \in \mathcal{C} \mid \gamma(c) > i\}$$

as the set of customers that have been treated before I_i.
Let $m \in C^{-1}\left(\overline{\mathcal{C}}_i\right)$ and let $\{S_{i+1}, \dots, S_n\}$ be the already determined partial solutions with $S_j = (x^j, z^j, e^j, u^j)$. We define

$$\hat{u}^i_m := u^j_m \qquad \text{for } j = (C \circ \gamma)(m).$$

In the partial instance I_i, we choose a route for each customer $c \in \mathcal{C}_i$. This is indicated by the decision variables u^i_m for $m \in C^{-1}(c)$. We save this decision for all partial instances that are solved later in the value $\hat{u}^j_m$ for $j \in [i-1]$. Thus, we are given a $\hat{u}^i_m \in \{0,1\}$ for each $m \in C^{-1}\left(\overline{\mathcal{C}}_i\right)$. We introduce the constraint

$$\sum_{s \in \mathrm{N}^-_{\overline{G}_i}(t)} x_{s,t} = \hat{u}^i_m \qquad \text{for all } m \in C^{-1}\left(\overline{\mathcal{C}}_i\right), t \in m \cap \mathcal{T}_i \tag{4.46}$$

which ensures that the previous route choices are considered.

Cost Estimation

We have to select one route of customer $c \in \mathcal{C}$ in the partial instance $I_{\gamma(c)}$. For this we have to estimate the costs for all routes $m \in C^{-1}(c)$ in advance. However, these costs partly depend on the later solved instances. The entire cost $\text{cost}(S)$ consists of the vehicle cost c^{v}, trip cost c^{t}, deadhead cost c^{d} and route cost c^{r}. While the trip cost and route cost can be determined easily for a route, the vehicle cost and deadhead cost strongly depend on the environment of the route and cannot be determined. We therefore focus on the trip and route cost and define the estimated route cost as follows:

$$C_1(m) := c^{\text{r}}_m + \sum_{t \in m} c^{\text{t}}_t \qquad \text{for } m \in \mathcal{M}$$

We use this cost in order to define the modified route cost

$$\hat{c}^{\text{r}}_m := c^{\text{r}}_m + \sum_{t \in m \setminus \mathcal{T}_i} c^{\text{t}}_t \qquad \text{for } i \in [n], m \in C^{-1}\left(\mathcal{C}_i\right).$$

With this, we state the objective function as follows: Let $m \in C^{-1}\left(\mathcal{C}_i\right)$. Its trips in the partial trip set $t \in m \cap \mathcal{T}_i$ are not considered in $\hat{c}^{\text{r}}_m$ since their trip costs are already part of the objective function. The other trips $t \in m \setminus \mathcal{T}_i$ are added to $\hat{c}^{\text{r}}_m$ such that they have an impact on the choice of the routes.
Let $m \in C^{-1}\left(\overline{\mathcal{C}}_i\right)$. Its trips in the considered partial trip set $t \in m \cap \mathcal{T}_i$ are respected in $\hat{c}^{\text{r}}_m$ in the partial instance $I_{\gamma(C(m))}$ and as trip costs c^{t} in I_i. Therefore we subtract these trip costs from the objective function in I_i. This term is only a constant and does not influence the solution. We subtract it nevertheless, such that the sum of the objective values of the partial instances provides the correct solution cost.
In summary, the objective function reads as:

$$\begin{aligned} &\sum_{s \in \mathcal{T}_i} x_{s,d^{\text{e}}} c^{\text{v}} + \sum_{t \in \mathcal{T}_i} x_{d^{\text{s}},t} c^{\text{t}}_t + \sum_{m \in C^{-1}(\mathcal{C}_i)} u_m \hat{c}^{\text{r}}_m - \sum_{m \in C^{-1}\left(\overline{\mathcal{C}}_i\right)} \hat{u}^i_m \left(\sum_{t \in m \cap \mathcal{T}_i} c^{\text{t}}_t \right) \\ &\sum_{t \in \mathcal{T}_i \cup \hat{\mathcal{P}}_i} \sum_{s \in \mathrm{N}^-_{G_i}(t) \setminus \{d^{\text{s}}\}} \left[x_{s,t} \left(c^{\text{d}}_{s,t} + c^{\text{t}}_t \right) + \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} \left(c^{\text{d}}_{s,r} + c^{\text{d}}_{r,t} - c^{\text{d}}_{s,t} \right) \right] \qquad (\text{TMILP}_i) \end{aligned}$$

Model Equivalence

Finally, we show that the original model and the time-dependent heuristic are equivalent to each other. We can transform each heuristical solution into a feasible solution of the (MMILP). We show this in Theorem 5 and Theorem 10. In contrast to the customer-dependent splitting, we additionally can transform each solution of the

(MMILP) into a solution that is feasible in the heuristic. We prove this in Lemma 4 and Theorem 11.

Theorem 10 (Transformation of (TMILP$_i$) into (SMILP))**.** *Let $\{S_1, \dots, S_n\}$ be partial solutions where S_i is feasible in the (TMILP$_i$) and the end point set $\hat{\mathcal{P}}_i$ is created according to Algorithm 1 for $i \in [n]$. They can be transformed into a solution $\overline{S}$ that is feasible in the (SMILP) and the values $\sum_{i=1}^n \operatorname{val}(S_i)$ and $\operatorname{val}(\overline{S})$ coincide.*

Proof. For $i \in [n]$, let $S_i = (x^i, z^i, e^i, u^i)$ be the partial solutions of I_i and $\hat{u}^i_m$ as in Definition 17. The formulation (TMILP$_i$) is built on the (SMILP$_i$) with additional constraints and thus S_i is feasible in (SMILP$_i$). According to Theorem 6, we can construct a solution $\overline{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ which is feasible in the (SMILP) without the cover constraints. We apply this construction and only modify the variables $\bar{u}_m$.
Let $i \in [n]$ be arbitrary. We define

$$\bar{u}_m := u^i_m \qquad\qquad \text{for } m \in C^{-1}(\mathcal{C}_i).$$

The variables $\bar{u}_m$ only affect the cover constraints. For each $c \in \mathcal{C}$, there is a unique $i \in [n]$ such that $c \in \mathcal{C}_i$. The constraint (4.43) holds in every partial instance, from this follows (3.2). Let $m \in C^{-1}(\mathcal{C}_i)$. The constraint (3.3) follows from (4.44) for $t \in m \cap \mathcal{T}_i$ and from (4.46) for $t \in m \cap \bigcup_{j=1}^{i-1} \mathcal{T}_j$. Therefore $\overline{S}$ is a feasible solution of the (SMILP). According to Theorem 6, the objective values coincide. We have modified the objective function of the (TMILP$_i$) and $\bar{u}_m$. We see that

$$\begin{aligned}
\sum_{m\in\mathcal{M}} \bar{u}_m c^{\mathrm{r}}_m &= \sum_{i=1}^n \sum_{m\in C^{-1}(\mathcal{C}_i)} u^i_m c^{\mathrm{r}}_m \\
&= \sum_{i=1}^n \sum_{m\in C^{-1}(\mathcal{C}_i)} u^i_m \left(c^{\mathrm{r}}_m + \sum_{t\in m\setminus\mathcal{T}_i} c^{\mathrm{t}}_t \right) - \sum_{i=1}^n \sum_{m\in C^{-1}(\overline{\mathcal{C}}_i)} \hat{u}^i_m \left(\sum_{t\in m\cap\mathcal{T}_i} c^{\mathrm{t}}_t \right) \\
&= \sum_{i=1}^n \left[\sum_{m\in C^{-1}(\mathcal{C}_i)} u^i_m \hat{c}^{\mathrm{r}}_m - \sum_{m\in C^{-1}(\overline{\mathcal{C}}_i)} \hat{u}^i_m \left(\sum_{t\in m\cap\mathcal{T}_i} c^{\mathrm{t}}_t \right) \right]
\end{aligned}$$

and therefore

$$\operatorname{val}\left(\overline{S}\right) = \sum_{i=1}^n \operatorname{val}(S_i).$$

□

Lemma 4. *Let S be a solution of the (MMILP). We can transform S into a solution $\overline{S}$ which is feasible in the (SMILP) with a splitting according to Definition 16. The objective values $\operatorname{val}(S)$ and $\operatorname{val}(\overline{S})$ coincide.*

Proof. Let $S = (x, z, e, u)$ be a feasible solution of the (MMILP). We construct a solution $\overline{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ that is feasible in the (SMILP) with a time-dependent splitting. Let $i \in [n]$ and $s \in \mathcal{T}_i$. For each $t \in \mathcal{T}$ with $s \prec t$ holds $t \in \bigcup_{j=i}^{n} \mathcal{T}_j$ due to the construction of the splitting.
For $s, t \in \mathcal{T}_i$ (if $i = 1$ then $s \in \mathcal{V} \cup \mathcal{T}_1$) with $s \prec t$ and $r \in \mathcal{R}_{s,t}$, we set

$$\bar{x}_{s,t} := x_{s,t} \qquad\qquad \bar{z}_{s,r,t} := z_{s,r,t}.$$

For $s \in \mathcal{T}_i$ (if $i = 1$ then $s \in \mathcal{V} \cup \mathcal{T}_1$), $t' = \mathrm{SP}_i(t)$ for some $t \in \bigcup_{j=i+1}^{n} \mathcal{T}_j$ with $s \prec t$ and $r \in \mathcal{R}_{s,t}$, we set

$$\bar{x}_{s,t'} := x_{s,t} \qquad\qquad \bar{x}_{t',t} := x_{s,t} \qquad\qquad \bar{z}_{s,r,t'} := z_{s,r,t}.$$

Further, we set:

$$\begin{array}{llllll} \bar{u}_m := u_m & \text{for } m \in \mathcal{M} & \bar{x}_{d^{\mathrm{s}},s} := x_{d^{\mathrm{s}},s} & \bar{x}_{s,d^{\mathrm{e}}} := x_{s,d^{\mathrm{e}}} & \text{for } s \in \mathcal{V} \cup \mathcal{T} \\ \bar{e}_s := e_s & \text{for } s \in \mathcal{V} \cup \mathcal{T} & \bar{e}_s := e_t + f_t^{\mathrm{t}} & \text{for } t \in \mathcal{T}, s = \mathrm{SP}_i(t) \end{array}$$

The proof that $\overline{S}$ is a feasible solution of the (SMILP) and the objective values coincide works analogously to the proof of Theorem 5. □

Theorem 11 (Transformation of (SMILP) into (TMILP$_i$))**.** *Let $\overline{S}$ be a feasible solution of the (SMILP). We can transform $\overline{S}$ into partial solutions $\{S_1, \dots, S_n\}$ where S_i is feasible in (TMILP$_i$) for $i \in [n]$. The values val$(\overline{S})$ and $\sum_{i=1}^{n} \mathrm{val}\,(S_i)$ coincide.*

Proof. Let $\overline{S} = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ be a feasible solution of the (SMILP). We construct solutions $S_i = \left(x^i, z^i, e^i, u^i\right)$ for $i \in [n]$. We define x^i, z^i and e^i as in the proof of Theorem 7. For $m \in C^{-1}(\mathcal{C}_i)$, we define

$$u_m^i := \bar{u}_m \qquad\qquad \hat{u}_m^j := \bar{u}_m \qquad\qquad \text{for } j \in [i-1].$$

The triple $\left(x^i, z^i, e^i\right)$ is feasible in the (SMILP) according to Theorem 7 and the cover constraints (4.43), (4.44), (4.45) and (4.46) follow directly from (3.2), (3.3) and (3.14). Therefore, each S_i is a feasible solution of (TMILP$_i$). The proof of

$$\sum_{i=1}^{n} \mathrm{val}\,(S_i) = \mathrm{val}\left(\overline{S}\right)$$

works analogously to the proof of Theorem 10. □

4.3.2 Iterative Approach

We use the previously developed heuristic for an iterative approach. First, we compute an initial solution while we choose the routes according to cost function C_1. Based on this solution, we determine the actual costs of the selected routes. With this, we can estimate the contribution of a route to the objective function. We compare the estimated route cost to the actual route cost. If the actual route cost is considerably higher than the estimated route cost, this route choice was likely bad. We identify the customers with the worst route choices and solve a subproblem, where we fix all route choices except for the considered customers. Regarding one customer set after another, we can iteratively improve the solution.

Initial Solution

We determine a solution with Algorithm 1 and a splitting according to Definition 16. Based on this solution $S = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$, we determine

$$C_1(c) := C_1(m) \qquad \text{for } c \in \mathcal{C}, m \in C^{-1}(c) \text{ with } \bar{u}_m = 1.$$

Finding Bad Route Choices

For an existing solution, the subproblem is to find a customer $\bar{c}$ with a bad route choice. This means, for this customer there is another route, such that the total cost decreases by choosing the other route. We can exchange these routes and compute a new solution considering the new route choice.
We cannot determine the cost saving for exchanging routes of a customer exactly without solving the complete problem again. Instead, we try to estimate the cost that a route contributes to the total cost. We can neither determine the contributing cost exactly. A simple deletion of the trips of the considered route does not necessarily yield the minimal cost, since it does not consider possible rearrangements of the schedule. Instead of computing the contributing cost exactly, we try to estimate the cost that is caused by this route in the solution. If this contributing cost are considerably higher than the previously estimated cost, this customer is a candidate for exchanging routes.
Let $S = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ be a feasible solution of the (MMILP). In order to determine the contributing cost for route $m \in \mathcal{M}$, we define the following auxiliary costs for every trip $t \in \mathcal{T}$ that is covered in the solution.

1. Vehicle cost $c_t^{\text{v}}(S)$: Let $v \in \mathcal{V}$ be the vehicle covering t and k_v the number of trips covered by v:

$$c_t^{\text{v}}(S) := \frac{c^{\text{v}}}{k_v}$$

The vehicle cost $c^{\mathrm{v}}_t(S)$ equally partitions the vehicle cost c^{v} of a duty to all of its trips.

2. Refueling cost $c^{\mathrm{refuel}}_t(S)$: Let $r \in \mathcal{R}$ be the next refuel station used after t and T_r all trips covered since the last station, let $\bar{z}_{s,r,s'} = 1$:

$$c^{\mathrm{refuel}}_t(S) := \frac{f^{\mathrm{t}}_t}{\sum_{t' \in T_r} f^{\mathrm{t}}_t} \left(c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,s'} - c^{\mathrm{d}}_{s,s'} \right)$$

If the vehicle is not refueled after t, then $c^{\mathrm{refuel}}_t(S) := 0$. The refueling cost $c^{\mathrm{refuel}}_t(S)$ treats the additional cost for refueling. For all trips that are covered between two refuel points, the cost for the refueling detour is divided to the trips according to their respective fuel consumptions.

3. Deadhead cost $c^{\mathrm{d}}_t(S)$: Let $s \in \mathcal{V} \cup \mathcal{T}, s' \in \mathcal{T}$ be the trips covered directly before and after t by vehicle v, i.e. $\bar{x}_{s,t} = \bar{x}_{t,s'} = 1$:

$$c^{\mathrm{d}}_t(S) := \frac{1}{2} \left(c^{\mathrm{d}}_{s,t} + c^{\mathrm{d}}_{t,s'} \right)$$

If t is the last trip of the duty, i.e. $\bar{x}_{s,t} = \bar{x}_{t,d^{\mathrm{e}}} = 1$, then $c^{\mathrm{d}}_t(S) := \frac{1}{2} c^{\mathrm{d}}_{s,t}$. The deadhead cost $c^{\mathrm{d}}_t(S)$ assigns the cost for the deadhead trips to the respective trips. For each deadhead trip, the deadhead cost $c^{\mathrm{d}}_{s,t}$ is equally divided up to s and t.

The schedule cost according to Definition 6 comprises vehicle costs, trip costs, deadhead costs and route costs. To each multimodal route, we can easily assign its route cost and the sum of its trip costs. In contrast, we cannot directly assign the vehicle and deadhead costs. Thus, we use the auxiliary costs in order to assign the vehicle and deadhead costs approximately to the trips and the routes. With these auxiliary costs we can define new route costs which give a better description of the contribution of a multimodal route to the entire solution:

Definition 18 (Improved Cost Estimation)**.** Let $S = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ be a solution of the (MMILP). With the auxiliary costs described in this paragraph, we define the improved cost estimation for all multimodal routes $m \in \mathcal{M}$ with $\bar{u}_m = 1$:

$$C_2(S, m) := C_1(m) + \sum_{t \in m} \left(c^{\mathrm{v}}_t(S) + c^{\mathrm{refuel}}_t(S) + c^{\mathrm{d}}_t(S) \right)$$

We further define

$$C_2(S, c) := C_2(S, m) \qquad \text{for } c \in \mathcal{C}, m \in C^{-1}(c) \text{ with } \bar{u}_m = 1.$$

Now we can evaluate the previous estimation for the route contribution. If $C_2(S, c)$ is significantly higher than $C_1(S, c)$ then possibly a bad route has been chosen for customer $c \in \mathcal{C}$.
After defining an indicator for bad route choices, we decide which of the customers we want to review. It is presumably not profitable to solve a subproblem for a single customer. Instead we create a set $B \subseteq \mathcal{C}$ with customers that are reviewed. In total, we aim to develop an efficient procedure. To achieve this, the subproblem should significantly decrease the objective value of the solution on the one hand, on the other hand it should be solved very fast. We examine desirable properties of the customer set B. Note that we provide only indicators for the properties and statements about the qualitative behavior.
Let $c \in \mathcal{C}$ with $\bar{m} \in C^{-1}(c)$ be the route chosen in the solution S. Assume that $\bar{m}$ is a bad route choice for c and with changing the route choice a large cost saving can be achieved. We explain qualitatively, how the respective costs behave. Then the following properties apply:

1. The ratio $\frac{C_2(S,c)}{C_1(\bar{m})}$ is large.
   Although $\bar{m}$ is a bad route choice, $C_1(\bar{m})$ is small enough that $\bar{m}$ was chosen in the original setting. $C_2(S, \bar{m})$ is large enough that the total cost decreases with choosing another route.

2. There are $m \in C^{-1}(c) \setminus \{\bar{m}\}$ with $C_1(m) < C_2(S, c)$.
   If $\bar{m}$ is a bad choice, better alternatives are available.

3. The difference $C_2(S, c) - C_1(\bar{m})$ is large.
   The cost that is saved by exchanging routes is bounded by this difference.

4. The set $(M \circ C)^{-1}(c) \setminus \mathcal{T}_{\gamma(c)}$ is large.
   This set contains all trips of c that are not in the partial instance where the route has been chosen. If this set is large, much information about these trips were not available and the route decision is heavily based on the cost estimations. Thus $\bar{m}$ is more likely a bad route choice.

We consider these properties for creating the customer set B. Further we have to keep the subproblem small, such that it can be solved in reasonable time. Therefore we make additional assumptions on B:

5. The time interval $[\max_{c \in B} z_c^{\text{end}}, \min_{c \in B} z_c^{\text{start}}]$ is small.
   In the subproblem, all duties stay fixed up to this time interval. A small time interval causes a small number of trips that are variable w. r. t. the duties. This keeps the size of the subproblem small.

6. The set $(M \circ C)^{-1}(B)$ is small.
   This set contains all trips of one of the customers in B. In the subproblem, all

other trips are fixed to be fulfilled. Therefore a small number of trips that are variable w. r. t. the duties or can be neglected keeps the subproblem small.

It is not possible to meet the requirements for B and simultaneously cover all customers with potential bad route choices. Therefore we partition the customer set into several subsets $B = \bigcup_{j=1}^{n} B_j$. It suffices if B_j fulfill the set requirements for all $j \in [k]$. We iteratively execute the subproblem, each with a small customer set B_j.

Algorithm 2: Determination of critical customers

```
   Input: solution $S$, $\mathcal{C}$, $\bar{m}_c$ for $c \in \mathcal{C}$, $r_{\min}, c_{\max}, t_{\max}, n_{\max}$
   Output: $\{B_1, \dots, B_k\}$
1  foreach $c \in \mathcal{C}$ do determine $C_2(S, c)$;
2  $B \leftarrow \left\{ c \in \mathcal{C} \mid \frac{C_2(S,c)}{C_1(\bar{m}_c)} \geq r_{\min}, \exists m \in C^{-1}(c) \setminus \{\bar{m}_c\} \text{ with } C_1(m) < C_2(S, c) \right\}$;
3  $i \leftarrow 1$;
4  while $B \neq \emptyset \wedge i \leq n_{\max}$ do
5      $\bar{c} \leftarrow \arg\max_{c \in B} \left( \frac{C_2(S,c)}{C_1(\bar{m}_c)} \right)$;
6      $B_i \leftarrow \left\{ c \in B \mid z_{\bar{c}}^{\text{start}} - \frac{t_{\max}}{2} \leq z_c^{\text{start}}, z_c^{\text{end}} \leq z_{\bar{c}}^{\text{start}} + \frac{t_{\max}}{2} \right\}$;
7      while $|B_i| > c_{\max}$ do $B_i \leftarrow B_i \setminus \left\{ \arg\min_{c \in B_i} \left( \frac{C_2(S,c)}{C_1(\bar{m}_c)} \right) \right\}$;
8      if $B_i = \emptyset$ then $B \leftarrow B \setminus \{\bar{c}\}$;
9      else $B \leftarrow B \setminus B_i$; $i \leftarrow i + 1$;
10 end
11 $k \leftarrow i$;
12 return $\{B_1, \dots, B_k\}$
```

Algorithm 2 shows how we create the customer sets. In order to meet the requirements, we introduce the following parameters:

- $r_{\min}$: The minimal ratio $\frac{C_2(S,c)}{C_1(\bar{m})}$ for $c \in B$
- $c_{\max}$: The maximal number of customers in B_j
- $t_{\max}$: The maximal time range of the customers in B_j
- $n_{\max}$: The maximal number of iterations

The properties taken into account by Algorithm 2 as follows: With the choice of B the amount of the ratio and the availability of alternatives is considered. By the choice of $c_{\max}$ and $t_{\max}$ the size of the subproblem is restricted. By iteratively choosing the customer with the highest ratio and restricting the number of iterations, we ensure that many customers with high potential for improvement are treated in a short time. After creating the sets of customers to review, we describe the subproblem for exchanging routes.

Subproblem

Let $S = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ be a solution of the (MMILP) and $B \subseteq \mathcal{C}$ a set of candidates for a bad route choice. We define the following subproblem (HSP$_B$): Assume the schedule according to S for the entire time without the trips of B and all route choices for customers except B to be fixed. Determine an optimal schedule within these restrictions.

Definition 19. Let $B \subseteq \mathcal{C}$. We set

$$z_B^{\text{start}} := \min_{c \in B} z_c^{\text{start}} \qquad\qquad z_B^{\text{end}} := \max_{c \in B} z_c^{\text{end}}$$

and define a splitting $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ by

$$\mathcal{T}_i := \begin{cases} \{t \in \mathcal{T} \mid z_t^{\text{start}} < z_B^{\text{start}}\} & \text{if } i = 1 \\ \left\{t \in \mathcal{T} \mid z_B^{\text{start}} \leq z_t^{\text{start}} \leq z_B^{\text{end}}\right\} & \text{if } i = 2 \\ \left\{t \in \mathcal{T} \mid z_B^{\text{end}} < z_t^{\text{start}}\right\} & \text{if } i = 3. \end{cases}$$

The (HSP$_B$) is a partial instance of the Successive Heuristic with a splitting according to Definition 19. We split the duties of S into three partial solutions $\{S_1, S_2, S_3\}$, each having trips only in $\mathcal{T}_i$. For this, we identify the transitions from $\mathcal{T}_1$ to $\mathcal{T}_2$ and from $\mathcal{T}_2$ to $\mathcal{T}_3$. These transitions form a start point set $\hat{\mathcal{V}}$ and an end point set $\hat{\mathcal{P}}$. We solve the subproblem with a modified formulation of (SMILP$_1$). The vehicle set is replaced by $\hat{\mathcal{V}}$ and the end point set is $\hat{\mathcal{P}}$.
The start point set comprises the respective last trips of all partial duties of S_1 and all vehicles that have no duty in this partial solution. The end point set comprises the respective first trips in S_3. We set $\hat{\mathcal{V}}$ and $\hat{\mathcal{P}}$ as

$$\begin{aligned} \hat{\mathcal{V}} &:= \{s \in \mathcal{V} \cup \mathcal{T}_1 \mid \bar{x}_{s,t} = 1 \text{ for some } t \in \mathcal{T}_2 \cup \mathcal{T}_3 \cup \{d^{\text{e}}\}\} \\ \hat{\mathcal{P}} &:= \{t \in \mathcal{T}_3 \mid \bar{x}_{s,t} = 1 \text{ for some } s \in \mathcal{V} \cup \mathcal{T}_2 \cup \mathcal{T}_3\}\,. \end{aligned}$$

Strictly speaking, $\hat{\mathcal{V}}$ and $\hat{\mathcal{P}}$ consist of start or end points that are created from the respective trips. For each start or end point t that is created from a trip or vehicle s hold the following properties:

$$\begin{aligned} p_t^{\text{start}} &= p_t^{\text{end}} := p_s^{\text{end}} & z_t^{\text{start}} &= z_t^{\text{end}} := z_s^{\text{end}} && \text{for all } t \in \hat{\mathcal{V}} \\ p_t^{\text{start}} &= p_t^{\text{end}} := p_s^{\text{start}} & z_t^{\text{start}} &= z_t^{\text{end}} := z_s^{\text{start}} && \text{for all } t \in \hat{\mathcal{P}} \end{aligned}$$

How the initial fuel states f_t^0 for $t \in \hat{\mathcal{V}} \cup \hat{\mathcal{P}}$ are created, is explained afterwards.

Using these definitions, we adapt the formulation (SMILP$_1$) to (HSP$_B$). The vehicle set $\mathcal{V}$ is replaced by the start point set $\hat{\mathcal{V}}$. We add the cover constraints:

$$\sum_{m \in C^{-1}(c)} u_m = 1 \qquad \text{for all } c \in B \tag{4.47}$$

$$\sum_{s \in \mathrm{N}^-_{\overline{G}_1}(t)} x_{s,t} = u_m \qquad \text{for all } m \in C^{-1}(B), t \in m \tag{4.48}$$

$$\sum_{s \in \mathrm{N}^-_{\overline{G}_1}(t)} x_{s,t} = \bar{u}_{M(t)} \qquad \text{for all } t \in \mathcal{T}_2 \setminus (M \circ C)^{-1}(B) \tag{4.49}$$

$$u_m \in \{0,1\} \qquad \text{for all } m \in C^{-1}(B) \tag{4.50}$$

where $\bar{u}_m$ are the fixed route decisions of S for $m \in \mathcal{M} \setminus C^{-1}(B)$. The constraints (4.47), (4.48) and (4.50) ensure the cover constraints for all trips $t \in (M \circ C)^{-1}(B)$, i. e. all trips of the reviewed customers. (4.49) ensures that the fixed route decisions for all other customers are maintained.

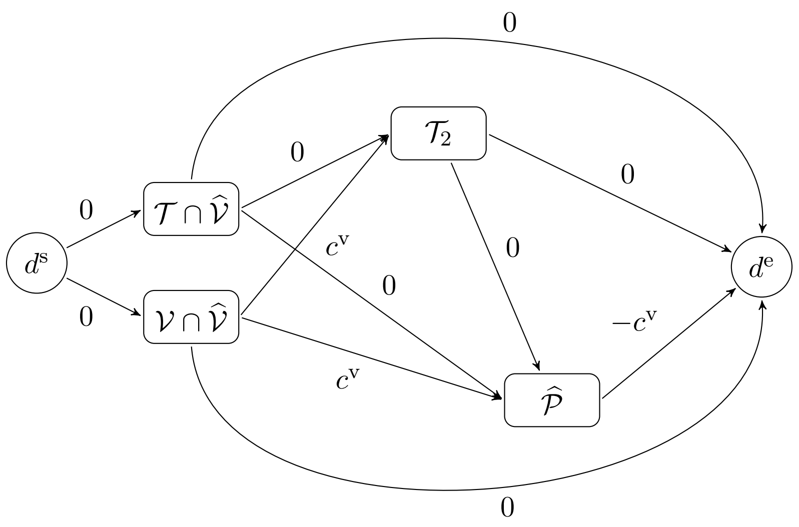

Figure 4.7: Vehicle cost in (HSP$_B$)

In (HSP$_B$), we decide the route of customers $c \in B$. Thus we add their route cost to the objective function. The vehicle cost is only caused by the vehicles, but not by the other start points. A partial duty that ends with an end point does not cause any vehicle cost since this cost is already considered in S_3. A duty that starts with a trip $s \in \mathcal{T} \cap \hat{\mathcal{V}}$ and ends with a trip $t \in \hat{\mathcal{P}}$ has negative vehicle cost since it has been charged in S_1 and S_3. Figure 4.7 illustrates the associated task graph and the vehicle cost for the subproblem. We replace the term

$$\left(\sum_{s \in \mathcal{V}} \sum_{t \in \mathrm{N}^{+}_{G_1} \setminus \{d^{\mathrm{e}}\}} x_{s,t} - \sum_{s \in \hat{\mathcal{P}}_1} x_{s,d^{\mathrm{e}}} \right) c^{\mathrm{v}}$$

in (SMILP$_1$) by

$$\left(\sum_{s \in \mathcal{V} \cap \hat{\mathcal{V}}} \sum_{t \in \mathrm{N}^{+}_{G_1} \setminus \{d^{\mathrm{e}}\}} x_{s,t} - \sum_{s \in \hat{\mathcal{P}}} x_{s,d^{\mathrm{e}}} \right) c^{\mathrm{v}} + \sum_{m \in C^{-1}(B)} u_m c^{\mathrm{r}}_m.$$

Determination of the Initial Fuel States

We now examine the determination of the initial fuel states f^0_t for $t \in \hat{\mathcal{V}}_2 \cup \hat{\mathcal{P}}_2$. These fuel states should guarantee a feasible continuation of the existing duties. Besides this, we aim to have as much scope as possible for feasible solutions. Therefore the initial fuel state should be large for a start point and small for an end point. In order to determine the optimal initial fuel states, we draw up another linear program where all results are fixed by the existing solution and only the fuel states stay variable. Then we optimize the respective fuel values.
Given a solution $S = (\bar{x}, \bar{z}, \bar{e}, \bar{u})$ that is feasible in the (MMILP). We state the problems (FMILP$^{\min}_S$) and (FMILP$^{\max}_S$) as follows:

$$\begin{aligned}
\min \quad & \sum_{s \in \mathcal{V} \cup \mathcal{T}} e_s && && (\mathrm{FMILP}^{\min}_S) \\
\max \quad & \sum_{s \in \mathcal{V} \cup \mathcal{T}} e_s && && (\mathrm{FMILP}^{\max}_S) \\
\text{s.t.} \quad & e_s \le f^0_s && \text{for all } s \in \mathcal{V} && (3.7) \\
& 0 \le e_s - \sum_{r \in \mathcal{R}_{s,t}} \bar{z}_{s,r,t} f^{\mathrm{d}}_{s,r} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^{-}_{G}(t) && (4.51) \\
& e_t \le 1 - f^{\mathrm{t}}_t - \sum_{r \in \mathcal{R}_{s,t}} \bar{z}_{s,r,t} f^{\mathrm{d}}_{r,t} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^{-}_{G}(t) && (4.52) \\
& e_t \le e_s - \bar{x}_{s,t} \left(f^{\mathrm{d}}_{s,t} + f^{\mathrm{t}}_t \right) - \sum_{r \in \mathcal{R}_{s,t}} \bar{z}_{s,r,t} \left(f^{\mathrm{d}}_{s,r} + f^{\mathrm{t}}_r + f^{\mathrm{d}}_{r,t} - f^{\mathrm{d}}_{s,t} \right) + (1 - \bar{x}_{s,t}) \\
& && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^{-}_{G}(t) && (4.53) \\
& e_s \in [0,1] && \text{for all } s \in V \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\} && (3.13)
\end{aligned}$$

These formulations provide the same solution as S with minimal or maximal fuel states, respectively. Let $e^{\min}$ be a solution of the ($\text{FMILP}_S^{\min}$) and $e^{\max}$ be a solution of the ($\text{FMILP}_S^{\max}$), then we define the initial fuel states as follows:

$$f_t^0 := e_t^{\max} \qquad \text{for all } t \in \hat{\mathcal{V}}_2 \qquad f_t^0 := e_t^{\min} \qquad \text{for all } t \in \hat{\mathcal{P}}_2$$

Creating an Improved Solution

After constructing the subproblem, we describe how we compose the old solution and the result of the subproblem to a new feasible solution. Let S be the current solution, $B \subseteq \mathcal{C}$ the set of critical customers and $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ the respective splitting. We create the partial solutions S_1 and S_3. Partial solution S_1 consists of the partial duties of S ending with a trip or vehicle representing a start point $t \in \hat{\mathcal{V}}_2$. Partial solution S_3 consists of the partial duties of S starting with a trip representing an end point $t \in \hat{\mathcal{P}}_2$. Let $\widehat{S}_2$ be a solution of (HSP_B). Each of its partial duties certainly starts with a start point $t \in \hat{\mathcal{V}}$. It ends with a start point, a trip or an end point. For each partial duty $d \in \widehat{S}_2$, delete its start point and prepend the partial duty $d' \in S_1$ that ends with the trip representing the start point to it. If d ends with an end point, delete the end point and append the partial duty $d' \in S_3$ that starts with the trip representing the end point. All duties that are created in this way are part of the new solution $\widehat{S}$.
The cost of the new solution is

$$\operatorname{cost}\left(\widehat{S}\right) = \operatorname{cost}(S_1) + \operatorname{cost}\left(\widehat{S}_2\right) + \operatorname{cost}(S_3) + \sum_{c \in \mathcal{C} \setminus B} c^{\mathrm{r}}_{m(c)}$$

where $c \mapsto m(c)$ describes the route decision for the previous solution S. We show that $\widehat{S}$ is again feasible and the cost does not increase.

Lemma 5. *Let S be a feasible solution of the* (MMILP) *and $B \subseteq \mathcal{C}$. Let $\widehat{S}$ be a solution that is computed as described in Section 4.3.2. Then $\widehat{S}$ is also feasible and*

$$\operatorname{cost}\left(\widehat{S}\right) \leq \operatorname{cost}(S).$$

Proof. The partial solutions S_1, $\widehat{S}_2$ and S_3 are connected in such a way that $\widehat{S}$ is feasible w. r. t. time. The initial fuel states in $\hat{\mathcal{V}}$ and $\hat{\mathcal{P}}$ are computed appropriately in order to maintain fuel feasibility. For $c \in B$, the cover constraints are ensured by (4.47), (4.48) and (4.50). By construction, $\widehat{S}$ fulfills exactly the same trips as S for $c \in \mathcal{C} \setminus B$.
The deadhead and trip cost are determined as before. From the construction of the objective function of (HSP_B) follows that the vehicle cost arise exactly once for each non-empty duty in $\widehat{S}$. The vehicle cost is charged for each duty starting with a vehicle

and not with a trip. It is subtracted for each duty ending with an end point. For $c \in B$, the route cost occurs in cost $\left(\widehat{S}_2\right)$ and for $c \in \mathcal{C} \backslash B$ in the additional term.
The original solution S_2 is a feasible solution of the (HSP_B) by construction. Therefore the total cost of $\widehat{S}$ is not greater than the total cost of S. □

Finally, we describe the complete Iterative Approach to create an improved solution as follows:

1. Create an initial solution S^0 with the time-dependent Successive Heuristic according to Algorithm 1
2. Based on S^0, determine the customer sets $\{B_1, \dots, B_k\}$ with Algorithm 2
3. For $i \in [k]$: Based on the previous solution S^{i-1}, create the subproblem (HSP_{B_i}). Solve the subproblem and create an improved solution S^i. These procedures are described in Section 4.3.2.
4. The solution S^k is the improved solution.

Due to Lemma 5, all solutions S^i are feasible solutions and the total cost iteratively increases. We may use each feasible solution of the (MMILP) as in initial solution S^0. Further, each choice of the customer sets is possible, even if some customer arises more than once in the customer sets.

Remarks

We provide several remarks concerning the developed heuristics and the results that can be expected or not.
Analogously to the customer-dependent heuristic, the time-dependent heuristic is not a constant-factor approximation. Example 4 directly applies to (TMILP), too. With the same choice of the time point c_1, the splitting $\{\mathcal{T}_1, \mathcal{T}_2\}$ is equal if it is performed according to Definition 16. Thus the heuristic solution can be arbitrarily bad compared to the optimal solution, even if all subproblems are solved to optimality.
In the iterative approach, there is not guarantee that an optimal solution is achieved. If we choose $B_1 = \mathcal{C}$, i. e. all customers are reviewed during one iteration, then we will receive an optimal solution. But the formulation $(\mathrm{HSP}_{\mathcal{C}})$ is equal to (MMILP) and it is expected that this formulation is too big to solve it in reasonable time for realistic problem sizes. Even if $\bigcup_{i=1}^{k} B_i = \mathcal{C}$, i. e. every customer is reviewed during the process, we cannot expect an optimal solution.
The previously discussed criteria, whether the route choice of a customer was bad, are only indications. It is not sure that a customer that fulfills all of them actually has a bad route choice. Further, changing the route for a customer does not necessarily improve the cost of the current solution, even if the new route is also chosen in an

optimal solution. The subproblem does not certainly fix other bad heuristical decisions besides the route choice, e. g. the assignment of trips to duties or the visitation of refuel points.
There are several ways to further improve the heuristics. However, in order to focus on the optimal approach in Chapter 5, we will only sketch these improvements. We further cannot estimate how the additional computation time behaves in proportion to the additional cost saving. But we provide the main ideas in order to improve the heuristical solutions.

- We can put additional effort in the choice of the time points. We aim to have only few customers that are represented in more than one partial trip set. Preferably, the customers are represented in only few partial instances in summary. Nevertheless, the time points should be distributed evenly over time such that the partial instances have more or less the same size. We can create a supplementary subproblem for choosing the time points in a desirable way.

- In partial instance I_i, it is not beneficial to choose a route with $m \cap \mathcal{T}_i = \emptyset$. If such a route is the most suitable one, the route choice can be left open. In the next partial instance, the route is chosen among all routes with $m \cap \mathcal{T}_i = \emptyset$. In this case it is not necessary to fix the route choice in I_i because there are no trips to cover in this partial instance. In the next partial instance, there is more additional information available in order to choose a better route.

4.4 Conclusion

Finally, we summarize the important results of Chapter 4. We do not expect to solve the (MMILP) for realistic instance sizes in reasonable time. Thus we develop heuristical solution methods. They are based on a splitting of the complete instance into several partial instances. These partial instances are solved successively in reversed order. For each starting trip of a partial duty, we create an end point with which another partial duty ends. In the first partial instance, we assign these partial instances to the vehicles. Connecting these partial duties provides a feasible overall solution. The great challenge of the heuristic is the realization of the cover constraints. We develop two different approaches for this:
In the customer-dependent splitting, all trips of one customer are in the same partial trip set. Therefore, we can apply the cover constraints for each customer separately in the respective partial instance. While this approach is very simple, the disadvantage is that a feasible solution of the (MMILP) may not be feasible in the heuristic, if a later trip is in an earlier partial trip set. On the other hand, if the customer extension is bounded by the splitting length and we extend the vehicle set suitably, for each solution there exists a heuristic-feasible solution with at most doubled cost.

In the time-dependent splitting, trips of the same customer may be in different partial trip sets. In order to ensure the cover constraints, we choose the choose the route definitively in the respective partial instance which is solved earliest. For choosing the routes properly, we develop a cost estimation of the routes. While this approach requires more effort for the cover constraints, this heuristic formulation is equivalent to the original formulation. We additionally use the iterative approach for revising bad route choices. We identify potentially bad customers and iteratively review their route choices with specific subproblems.
For both heuristic approaches, we cannot provide a bound for the heuristic solution. There are examples in which the heuristic solution is arbitrarily bad compared to the optimal solution.
We use the heuristic solution in order to provide an initial solution for the Optimal Approach. This approach is examined in Chapter 5.

Chapter 5

Optimal Approach

In this chapter, we develop a solution method in order to solve our problem to optimality. We expect this to require a very efficient algorithm and a lot of computation power since the problem is $\mathcal{NP}$-hard. The solution method should cope with multi-leg cover constraints as defined in Chapter 2. The approach is based on the underlying master theses which have already found methods to solve a simplified version of our problem. [Kai16] provide an optimal algorithm for the problem with single-leg cover constraints. This problem setting assumes that each customer has a set of alternative trips, where one of them has to be fulfilled each. We want our algorithm to produce a result in reasonable time, therefore we require already a very good solution as an initial solution. For receiving a good initial solution, we apply the Successive Heuristics as developed in Chapter 4.

In order to tackle the problem, we introduce a path flow formulation which is different to the arc flow formulation of Section 3.2. Since the entire solution consists of separate vehicle duties, we decompose the problem into these single duties. This concept is an application of Dantzig-Wolfe Decomposition. There are only a few constraints connecting these subproblems, namely the cover constraints. First we regard the LP relaxation of this problem and solve this via column generation. The resulting subproblem for each duty is a shortest path problem with resource constraints (SPPRC), which is also $\mathcal{NP}$-hard. For solving this subproblem, we have both a heuristic and an exact algorithm. In order to receive a total solution, we apply branch-and-price. We provide and discuss some branching strategies used for this procedure.

Most of the procedure is already developed by [Kai16]. We show the crucial results for the algorithm and discuss our adaptions in the path flow formulation, the algorithm solving the subproblems and the branch-and-price procedure. This adaptions make the optimal approach also cope with multi-leg cover constraints.

5.1 Path Flow Formulation

We apply Dantzig-Wolfe Decomposition in order to create a path flow formulation of our problem. This is advantageous since the size of the arc flow formulation grows very fast with increasing problem size. We give only a short outline on the general procedure and then show the application to our problem.

5.1.1 Dantzig-Wolfe Decomposition

Dantzig-Wolfe Decomposition can be used in order to deal with large mixed-integer linear programs. It breaks the problem into smaller subproblems if the structure is suitable. This is the case if a large subset of the variables can be partitioned in a way such that the sets of occurring variables are disjoint for most of the constraints. The structure of the matrix for such a linear program looks as follows:

$$\left(\begin{array}{cccccccccccc}
\star & \cdots & \star & \star & \cdots & \star & & & & \star & \cdots & \star \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \cdots & & \vdots & \ddots & \vdots \\
\star & \cdots & \star & \star & \cdots & \star & & & & \star & \cdots & \star \\
\hline
\star & \cdots & \star & 0 & \cdots & 0 & & & & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \cdots & & \vdots & \ddots & \vdots \\
\star & \cdots & \star & 0 & \cdots & 0 & & & & 0 & \cdots & 0 \\
0 & \cdots & 0 & \star & \cdots & \star & & & & & & \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \ddots & & & \vdots & \\
0 & \cdots & 0 & \star & \cdots & \star & & & & & & \\
 & & & & & & & & & 0 & & 0 \\
 & \vdots & & & \ddots & & & \ddots & & \vdots & \ddots & \vdots \\
 & & & & & & & & & 0 & & 0 \\
0 & \cdots & 0 & & & & 0 & \cdots & 0 & \star & \cdots & \star \\
\vdots & \ddots & \vdots & & \cdots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & & & & 0 & \cdots & 0 & \star & \cdots & \star
\end{array}\right)$$

The entries with $\star$ denote the non-zero entries in the matrix.
The subproblems emerge by considering only the constraints of a single set of this partition. The other constraints concerning the whole variable set are called linking constraints as they link the respective subproblems. The master problem considers the objective function in connection with the linking constraints. We then apply column generation for each of the subproblems separately. Starting with only a small set of feasible solutions, we successively generate further feasible solutions and include them to the master problem. Each feasible solution represents a column of the matrix representing the linear program. In the master problem, the actual formulation of the subproblems is not needed. Thus, we can extract the subproblems and solve them with specialized algorithms if they have an appropriate structure.
The column generation method is only able to solve a linear program. Thus, we have to restate the integrality afterwards. How this is done is discussed later.

5.1.2 Application of the Decomposition

In the original problem formulation, we regard only a single set of variables which models the entire flow of the vehicles. For the arc flow formulation, a single variable set is advantageous as the corresponding task graph stays small. In contrast, we extend the variable set here in order to define smaller subproblems.

Identification of the Subproblems

Consider a solution of the (MMILP). This solution can be decomposed into a set of separate vehicle duties. For each of these duties, the time and fuel restrictions are applied individually. The only requirements that do not occur in the respective duties individually are the cover constraints. They guarantee that for each customer exactly one route is fulfilled and for each route, if it is fulfilled, each of its trips is fulfilled. Therefore the duty of each vehicle is a natural choice for the subproblems. We introduce (x^v, z^v, e^v) for $v \in \mathcal{V}$ as the specific variables for each vehicle. With this we define the set of feasible vehicle duties as X_v for $v \in \mathcal{V}$.

$$
\begin{aligned}
X_v := \Big\{ (x,z,e) \in \{0,1\}^A \times \{0,1\}^{(A \cap (\mathcal{V} \cup \mathcal{T})^2) \times \mathcal{R}} \times [0,1]^{\mathcal{V} \cup \mathcal{T}} \mid \\
& \sum_{t \in \mathrm{N}^-_G(s)} x_{t,s} = \sum_{t \in \mathrm{N}^+_G(s)} x_{s,t} && \text{for all } s \in V \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\} && (3.4)\\
& \sum_{s \in \mathrm{N}^-_G(v)} x_{s,v} = 1 && && (5.1)\\
& \sum_{s \in \mathrm{N}^-_G(t)} x_{s,t} = 0 && \text{for all } t \in \mathcal{V} \setminus \{v\} && (5.2)\\
& \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} \le x_{s,t} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) && (3.6)\\
& e_s \le f^0_s && \text{for all } s \in \mathcal{V} && (3.7)\\
& 0 \le e_s - \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{s,r} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) && (3.8)\\
& e_t \le 1 - f^{\mathrm{t}}_t - \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{r,t} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) && (3.9)\\
& e_t \le e_s - x_{s,t}\left(f^{\mathrm{d}}_{s,t} + f^{\mathrm{t}}_t\right) - \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t}\left(f^{\mathrm{d}}_{s,r} + f^{\mathrm{t}}_r + f^{\mathrm{d}}_{r,t} - f^{\mathrm{d}}_{s,t}\right) + (1 - x_{s,t}) \\
& && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) && (3.10)\\
& \Big\}
\end{aligned}
$$

Constraints (5.1) and (5.2) ensure that exactly vehicle v is used in this formulation.

We denote the set of feasible duties for any vehicle by $X := \bigcup_{v\in\mathcal{V}} X_v$. Any feasible solution of (MMILP) can be decomposed into vehicle duties. This is guaranteed by (3.5) which forces the duties of the vehicles to be disjoint with respect to the trips. The only variables that are not considered in X_v are the route variables u_m which can be determined by the arc variables $x_{s,t}$. The objective function is additive with respect to the decomposition except for the route cost which we then consider explicitly. We write the cost for a configuration (x^v, z^v, e^v) as $g(x^v, z^v, e^v)$.

The only constraints that are not ensured in X_v are the cover constraints (3.2) and (3.3). These are the linking constraints for the various subproblems. In summary, we can rewrite (MMILP) as:

$$
\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}} g(x^v, z^v, e^v) + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m & & \\
\text{s.t.} \quad & \sum_{m\in C^{-1}(c)} u_m = 1 & \text{for all } c \in \mathcal{C} & \qquad (3.2)\\
& \sum_{v\in\mathcal{V}} \sum_{s\in \mathrm{N}^-_G(t)} x^v_{s,t} = u_m & \text{for all } m \in \mathcal{M}, t \in m & \qquad (5.3)\\
& (x^v, z^v, e^v) \in X_v & \text{for all } v \in \mathcal{V} & \\
& u_m \in \{0,1\} & \text{for all } m \in \mathcal{M} & \qquad (3.14)
\end{aligned}
$$

Reduction of the Master Problem

Because of the introduction of variables for each vehicle, the resulting problem size is very large. For maintaining the master problem, not all information of X_v are needed. In order to fulfill (5.3) we need only the term $\sum_{s\in\mathrm{N}^-_G(t)} x^v_{s,t}$, which is the set of trips served by a specific vehicle. Therefore, we define the linear mapping:

$$
\psi : X \to \{0,1\}^{\mathcal{T}} \qquad\qquad (x, z, e) \mapsto \left(\sum_{s\in\mathrm{N}^-_G(t)} x_{s,t} \right)_{t\in\mathcal{T}}
$$

The dimension of the codomain of ψ is much smaller than the dimension of the domain. We can rewrite (MMILP) by using $y^v := \psi(x^v, z^v, e^v)$:

$$
\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}} \min g\left(\psi^{-1}\left(y^v\right)\cap X_v\right) + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m \\
\text{s.t.} \quad & \sum_{m\in C^{-1}(c)} u_m = 1 && \text{for all } c\in\mathcal{C} && (3.2)\\
& \sum_{v\in\mathcal{V}} y^v_t = u_m && \text{for all } m\in\mathcal{M}, t\in m && (5.4)\\
& y^v \in \psi\left(X_v\right) && \text{for all } v\in\mathcal{V} \\
& u_m \in \{0,1\} && \text{for all } m\in\mathcal{M} && (3.14)
\end{aligned}
$$

The mapping ψ is not injective in general. Thus, there is more than one feasible duty that serves exactly the trips of y^v. These duties can have different cost. We therefore use the minimal resulting cost

$$\min g\left(\psi^{-1}\left(y^v\right)\cap X_v\right) = \min\left\{g\left(x^v\right) | x^v\in X_v, \psi\left(x^v\right) = y^v\right\}.$$

This is the smallest cost of a vehicle duty that serves exactly the trips as indicated by the incidence vector y^v. We do not have to determine these costs now. As we will see later, the costs are a byproduct of solving the subproblems.

Column Generation

We apply column generation to our problem. For every $v\in\mathcal{V}$, let $\mathcal{I}_v$ be an index set for the finitely many points in $\psi\left(X_v\right)$ and let the columns of $Y^v\in\mathbb{R}^{\mathcal{T}\times\mathcal{I}_v}$ be exactly those points. Let $G^v\in\mathbb{R}^{1\times\mathcal{I}}$ be the respective values of $\min g\left(\psi^{-1}(\cdot)\cap X_v\right)$. Then we can reformulate the master problem as:

$$
\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}} G^v\lambda^v + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m && && \text{(IMP)}\\
\text{s.t.} \quad & \sum_{v\in\mathcal{V}} Y^v_{t,\cdot}\lambda^v = u_m && \text{for all } m\in\mathcal{M}, t\in m \\
& \sum_{m\in C^{-1}(c)} u_m = 1 && \text{for all } c\in\mathcal{C} \\
& \sum_{i\in\mathcal{I}_v} \lambda^v_i = 1 && \text{for all } v\in\mathcal{V} \\
& \lambda^v \in \{0,1\}^{\mathcal{I}_v} && \text{for all } v\in\mathcal{V} \\
& u_m \in \{0,1\} && \text{for all } m\in\mathcal{M}
\end{aligned}
$$

Then we regard the LP-relaxation of (IMP) by dropping the integrality constraints:

$$
\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}} G^v\lambda^v + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m && && \text{(LMP)}\\
\text{s.t.} \quad & \sum_{v\in\mathcal{V}} Y^v_{t,\cdot}\lambda^v = u_m && \text{for all } m\in\mathcal{M}, t\in m\\
& \sum_{m\in C^{-1}(c)} u_m = 1 && \text{for all } c\in\mathcal{C}\\
& \sum_{i\in\mathcal{I}_v} \lambda^v_i = 1 && \text{for all } v\in\mathcal{V}\\
& \lambda^v \in \mathbb{R}^{\mathcal{I}_v}_{\geq 0} && \text{for all } v\in\mathcal{V}\\
& u_m \geq 0 && \text{for all } m\in\mathcal{M}
\end{aligned}
$$

As a next step, we reduce the size of the problem by considering only subsets $\mathcal{J}_v \subset \mathcal{I}_v$ of the feasible solutions for all $v\in\mathcal{V}$ and formulate the relaxed restricted master problem:

$$
\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}} G^v_{\mathcal{J}_v}\lambda^v + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m && && \text{(LRMP)}\\
\text{s.t.} \quad & \sum_{v\in\mathcal{V}} Y^v_{t,\mathcal{J}_v}\lambda^v = u_m && \text{for all } m\in\mathcal{M}, t\in m\\
& \sum_{m\in C^{-1}(c)} u_m = 1 && \text{for all } c\in\mathcal{C}\\
& \sum_{i\in\mathcal{J}_v} \lambda^v_i = 1 && \text{for all } v\in\mathcal{V}\\
& \lambda^v \in \mathbb{R}^{\mathcal{J}_v}_{\geq 0} && \text{for all } v\in\mathcal{V}\\
& u_m \geq 0 && \text{for all } m\in\mathcal{M}
\end{aligned}
$$

Finally, we regard the dual relaxed restricted master problem. For this, we introduce dual variables $\gamma\in\mathbb{R}^{\mathcal{T}}$, $\mu\in\mathbb{R}^{\mathcal{V}}$ and $\eta\in\mathbb{R}^{\mathcal{C}}$. The dual problem is:

$$
\begin{aligned}
\max \quad & \sum_{c\in\mathcal{C}} \eta_c + \sum_{v\in\mathcal{V}} \mu_v && && \text{(DLRMP)}\\
\text{s.t.} \quad & \sum_{t\in\mathcal{T}} Y^v_{t,i}\gamma_t + \mu_v \leq G^v_i && \text{for all } v\in\mathcal{V}, i\in\mathcal{J}_v && (5.5)\\
& \eta_{C(m)} - \sum_{t\in m} \gamma_t \leq c^{\mathrm{r}}_m && \text{for all } m\in\mathcal{M} && (5.6)\\
& \gamma\in\mathbb{R}^{\mathcal{T}}\\
& \mu\in\mathbb{R}^{\mathcal{V}}\\
& \eta\in\mathbb{R}^{\mathcal{C}}
\end{aligned}
$$

5.1.3 Solving the Relaxed Master Problem

The size of the index set $\mathcal{I}_v$ of all feasible solutions of X_v can be exponential in the input size. Therefore, the formulation (IMP) is hard, even the relaxed version (LMP) is hard. In order to tackle the problem, we first consider a small subset $\mathcal{J}_v \subset \mathcal{I}_v$ of the index set. We solve the problem (LRMP) where only duties from the restricted set are allowed. Since $\mathcal{J}_v$ is small, it is easier to solve the problem. Originating from this solution, we iteratively enlarge $\mathcal{J}_v$ and solve (LRMP) until the solution is an optimal solution of (LMP). For this method arise the following questions.

1. Does this procedure come up with an optimal solution in finitely many steps?
2. How do we find columns to add?
3. How do we check for optimality in (LMP)?

If we iteratively add columns to $\mathcal{J}_v$, we finally have $\mathcal{J}_v = \mathcal{I}_v$ after a finite number of steps since $\mathcal{I}_v$ is a finite set. When this is reached, the problems (LRMP) and (LMP) are equivalent and thus the solution is optimal. Obviously, this behavior is not desirable as we do not want to solve the unrestricted problem. Thus, we hope to receive an optimal solution earlier.
We can check for optimality and find columns to add by using the dual problems of the restricted and the unrestricted problem. Consider a solution $(\lambda^v)_{v\in\mathcal{V}}$ of (LRMP) and its corresponding dual solution $(\gamma^*, \mu^*, \eta^*)$ which is feasible in (DLRMP). We want to check whether $(\lambda^v)_{v\in\mathcal{V}}$ is an optimal solution of the unrestricted problem (LMP). Due to strong duality, this is the case if and only if $(\gamma^*, \mu^*, \eta^*)$ is feasible in the unrestricted dual problem (DLMP).
We therefore consider the constraints of the dual problems. Constraints (5.6) are equivalent in both formulations. The constraints (5.5) read as follows in the unrestricted case

$$\sum_{t\in\mathcal{T}} Y^v_{t,i}\gamma_t + \mu_v \leq G^v_i \qquad \text{for all } v \in \mathcal{V}, i \in \mathcal{I}_v. \tag{5.7}$$

Since $(\gamma^*, \mu^*, \eta^*)$ is a solution of the restricted problem, (5.7) is fulfilled for all $i \in \mathcal{J}_v$. It remains to check $\mathcal{I}_v \backslash \mathcal{J}_v$ which leads to the following subproblem:

$$\text{Find } i \in \mathcal{I}_v\backslash\mathcal{J}_v \qquad \text{s.t.} \qquad \sum_{t\in\mathcal{T}} Y^v_{t,i}\gamma^*_t + \mu^*_v > G^v_i \qquad \text{for } v \in \mathcal{V} \tag{5.8}$$

Identification of the Subproblem

Recall the definitions $G_i^v = \min g\left(\psi^{-1}\left(Y_i^v\right) \cap X_v\right)$ and $Y_i^v = \psi(x,z,e)$ for the respective $(x,z,e) \in X_v$. Using this, we can rewrite the subproblem to:

$$\begin{aligned} \min \quad & g\left(x,z,e\right) - \sum_{t\in\mathcal{T}} \sum_{s\in \mathrm{N}_G^-(t)} x_{s,t}\gamma_t^* - \mu_v^* \\ \text{s.t.} \quad & (x,z,e) \in X_v \end{aligned} \tag{SP$_v$}$$

Actually, the term $g(x,z,e)$ would be $\min g\left(\psi^{-1}\left(\psi(x,z,e)\right) \cap X_v\right)$. The following result shows that this distinction is not necessary as this is done implicitly be solving the subproblem.

Lemma 6. *For $v \in \mathcal{V}$, an optimal solution $(x^*,z^*,e^*) \in X_v$ to the subproblem (SP$_v$) fulfills*

$$g\left(x^*,z^*,e^*\right) = \min g\left(\psi^{-1}\left(\psi\left(x^*,z^*,e^*\right)\right) \cap X_v\right).$$

In other words, the duty (x^*,z^*,e^*) has the smallest possible cost under all duties which serve the same set of trips.

This lemma is proven by [Kai16, pp. 42-43] and holds for our case, too.
How the subproblem (SP$_v$) is solved, is shown in Section 5.2. As mentioned before, the cost G_i^v are also determined by solving the subproblem. We receive a solution (x,z,e) of (SP$_v$) for all $v \in \mathcal{V}$ and simultaneously the cost $g(x,z,e)$. If we then add the corresponding duty to $\mathcal{J}_v$, we can easily use the determined cost for G^v.

Updating the Index Set

The value of a duty as determined in the subproblem is called reduced cost. As long as there exists a violated constraint in the dual problem, there exists a column with negative reduced cost. This is used for deciding if a duty is added to the index set. First, we solve the (DLRMP) and receive a solution (γ^*,μ^*,η^*). With this we solve the subproblems (SP$_v$) for all $v \in \mathcal{V}$ and receive solutions (x^v,z^v,e^v). We know that all of these duties with negative reduced cost correspond to a violated constraint in (DLMP). Thus we consider these duties in the next step.
For each $v \in \mathcal{V}$, we have a solution $(x^v,z^v,e^v) \in X_v$. This solution corresponds to an index $i \in \mathcal{I}_v$. If the reduced cost for the solution is negative, i. e. $\operatorname{val}\left(x^v,z^v,e^v\right) < 0$, then we update the index set as follows:

$$\mathcal{J}_v \leftarrow \mathcal{J}_v \cup \{i\} \qquad Y_{\cdot,i}^v \leftarrow \psi\left(x^v,z^v,e^v\right) \qquad G_i^v \leftarrow g\left(x^v,z^v,e^v\right)$$

If $\operatorname{val}\left(x^v,z^v,e^v\right) \geq 0$ for all $v \in \mathcal{V}$, then the dual solution (γ^*,μ^*,η^*) is feasible in the (DLMP) and the corresponding primal solution $(\lambda^v)_{v\in\mathcal{V}}$ is an optimal solution of the relaxed master problem (LMP).

Initial Solution

For starting the column generation method, an initial index set is required for $\mathcal{J}_v$ for all $v \in \mathcal{V}$. These index sets have to be feasible, i. e. each occurring duty is feasible and there is a solution satisfying the cover constraints (3.2) and (3.3) using only duties out of $\bigcup_{v \in \mathcal{V}} \mathcal{J}_v$. Otherwise the restricted problem is infeasible and its dual problem is unbounded. Then we do not receive a solution (x^*, z^*, e^*) of (DLRMP) with which we define the subproblems. As an initial solution we use a heuristical solution of the problem, as we have developed in Chapter 4.
Note that this procedure only provides a solution for the LP-relaxation of the master problem. In Section 5.3 we show how we receive a solution of the (IMP).

5.2 Solving the Subproblems

In every step of the master problem, we solve the subproblem (SP_v) for each vehicle $v \in \mathcal{V}$. This subproblem is equivalent to the Shortest Path Problem with Resource Constraints (SPPRC). A vehicle duty is expressed as a d^{s}-d^{e}-path whose first vertex is the respective vehicle v. The main resource is the fuel state of the vehicle, where refuel stations a have negative fuel consumption. The goal is to find a feasible path with negative reduced cost which is then added to the index set. Besides fuel, we use additional resources in order to solve the subproblem.

5.2.1 Shortest Path Problem with Resource Constraints

In this section, we summarize the crucial results for solving the subproblem optimally. For this, we use a label-setting algorithm which solves the (SPPRC) efficiently. The respective definitions and the algorithm are shown in detail in [Kai16] and [ID05]. The (SPPRC) is a generalization of the Shortest Path Problem and is $\mathcal{NP}$-hard, as shown in [HZ80, p. 307]. The problem is given by a graph, a set of resources and a relation on every arc that specifies the change of resources along its way.

Definition 20 (Graph with resource constraints)**.** We call $H := (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$ a graph with resource constraints for a set of resources $\mathcal{U}$ if

1. (V_H, A_H) is a directed graph with vertex set V_H and arc set A_H.
2. $\sqsubseteq \in \{\leq, =, \geq\}^{\mathcal{U}}$ is a vector of resource relations and is called the resource dominance relation. For two resources $r, \tilde{r} \in \mathbb{R}^{\mathcal{U}}$, we write $r \sqsubseteq \tilde{r}$ if $r_u \sqsubseteq_u \tilde{r}_u$ for all $u \in \mathcal{U}$ and say that $\tilde{r}$ dominates r. The subset of maximal vectors of a set $R \subseteq \mathbb{R}^{\mathcal{U}}$ with respect to $\sqsubseteq$ is denoted by
$$\max_{\sqsubseteq} R := \{r \in R \mid \forall \tilde{r} \in R : r \sqsubseteq \tilde{r} \Rightarrow r = \tilde{r}\}.$$

The closed cone of resource vectors less than or equal to zero with respect to $\sqsubseteq$ is denoted by

$$\mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0} := \left\{ r \in \mathbb{R}^{\mathcal{U}} \mid r \sqsubseteq 0_{\mathcal{U}} \right\}.$$

3. $I \subseteq \mathbb{R}^{\mathcal{U}}$ is the Cartesian product of closed intervals of $\mathbb{R}$. The projection onto a single resource $u \in \mathcal{U}$ denoted by $\Pi_u(I)$ is called its resource window. If $\Pi_u(I) = \mathbb{R}$ for some resource $u \in \mathcal{U}$ it is called unrestricted.

4. $\mathrm{REF} = (\mathrm{REF}_{u,w})_{(v,w)\in A_H}$ is a vector of binary relations $\mathrm{REF}_{v,w} \subseteq I \times I$ for all $(v,w) \in A_H$ such that the set of vectors related to some $r^v \in I$

$$\mathrm{REF}_{v,w}(r^v) := \{r^w \in I \mid (r^v, r^w) \in \mathrm{REF}_{v,w}\}$$

is closed, has a finite set of maximal vectors $\max_{\sqsubseteq} \mathrm{REF}_{v,w}(r^v)$ and fulfills

$$\forall r^w, \tilde{r}^w \in I, r^w \sqsubseteq \tilde{r}^w : \tilde{r}^w \in \mathrm{REF}_{v,w}(r^v) \Rightarrow r^w \in \mathrm{REF}_{v,w}(r^v).$$

$\mathrm{REF}_{v,w}$ is called the resource extension function with respect to $\sqsubseteq$ on the arc $(v,w) \in A_H$.

The resource vectors are assigned to the vertices of the graph. They describe the absolute amount of available resources at that vertex.
The resource dominance relation is a partial order on $\mathbb{R}^{\mathcal{U}}$. If two resource vectors are comparable, then the dominating vector is always preferable to the other. If it is desirable to have a high quantity of a resource, the resource relation is set to $\leq$, e. g. for the fuel resource. Otherwise it is set to $\geq$, e. g. for the modeling cost resource. If no general relation holds, it is set to $=$. The resource extension function models the change of the resource vectors along the arcs. It relates a resource vector to all possible outcomes when traveling along this arc.

Definition 21 (Monotone resource extension function)**.** A resource extension function $\mathrm{REF}_{v,w} \subseteq I \times I$ with respect to $\sqsubseteq$ is called monotone if

$$\forall r^v, \tilde{r}^v \in I, r^v \sqsubseteq \tilde{r}^v : \mathrm{REF}_{v,w}(r^v) \subseteq \mathrm{REF}_{v,w}(\tilde{r}^v)$$

holds.

The monotonicity is important for the consistency. If a resource vector is dominated by another, then there are not more possible outcomes than for the dominating one.
After introducing the graph with resource constraints, we define resource-feasible paths on this graph.

Definition 22 (Resource-feasible path)**.** Let $H = (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$ be a graph with resource constraints. A path $P := (v_0, \ldots, v_n)$ of length $n \in \mathbb{N}_0$ in H is called resource-feasible if

$$\exists r^{v_i} \in I, i \in \{0, \ldots, n\} : (r^{v_{i-1}}, r^{v_i}) \in \mathrm{REF}_{v_{i-1}, v_i} \; \forall i \in \{1, \ldots, n\}$$

holds. We say $(r^v)_{v \in P}$ witnesses resource-feasibility of P.

The witnessing resource vectors $(r^v)_{v \in \mathcal{P}}$ are the resources along this path, e. g. the fuel state of the respective trips of a vehicle duty.

Contraction and Inversion

We define the actions „contraction of an arc“ and „inversion of a graph“ in order to apply them to our problem.

Definition 23 (Contraction)**.** Let $H = (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$ be a graph with resource constraints, $(v, w) \in A_H$ be the only arc leaving some vertex $v \in V_H$ and $\mathrm{REF}_{v,w}$ be monotone.

1. For $(u, v) \in A_H$, the concatenation of resource extension functions is defined as

$$\begin{aligned} \mathrm{REF}_{v,w} \circ \mathrm{REF}_{u,v} := \{ \, (r^u, r^w) \in I \times I \mid \exists r^v \in I : \\ (r^u, r^v) \in \mathrm{REF}_{u,v} \wedge (r^v, r^w) \in \mathrm{REF}_{v,w} \} \, . \end{aligned}$$

2. The graph with resource constraints $\widehat{H} := \left(V_{\widehat{H}}, A_{\widehat{H}}, \sqsubseteq, I, \widehat{\mathrm{REF}} \right)$ which results from H by contracting the arc (v, w) is defined by the vertex set $V_{\widehat{H}} := V_H \backslash \{v\}$, the arc set

$$A_{\widehat{H}} := \left(A_H \cap V_{\widehat{H}}^2 \right) \cup \{ (u, w) \mid (u, v) \in A_H \} \, ,$$

and the resource extension function $(\mathrm{REF}_a)_{a \in A_{\widehat{H}}}$, where

$$\widehat{\mathrm{REF}}_a := \begin{cases} \mathrm{REF}_{u,w} \cup (\mathrm{REF}_{v,w} \circ \mathrm{REF}_{u,v}) & \text{if } \exists u \in V_{\widehat{H}} : a = (u, w) \\ \mathrm{REF}_a & \text{otherwise} \end{cases}$$

for all $a \in A_H$. $\mathrm{REF}_{u,w}$ and $\mathrm{REF}_{u,v}$ are considered to be the empty relation $\emptyset$ if $(u, w) \notin A_H$ or $(u, v) \notin A_H$, respectively.

In [Kai16, p. 79] is proven that for a resource-feasible path P in the original graph H there is a resource-feasible path $\widehat{P}$ in the contracted graph $\widehat{H}$ and vice versa such that P and $\widehat{P}$ cover the same vertices of $V_H \backslash \{v\}$. We need contraction since we have resources on both the vertices and the arcs in our problem.

Definition 24 (Inversion)**.** 1. A resource extension function $\mathrm{REF}_{v,w}$ with respect to $\sqsubseteq$ is called invertible if the inverted relation

$$\mathrm{REF}_{v,w}^{-1} := \{(r^w, r^v) \mid (r^v, r^w) \in \mathrm{REF}_{v,w}\}$$

is a resource extension function with respect to the inverted dominance relation $\sqsupseteq$. $\mathrm{REF}_{v,w}^{-1}$ is called the inversion of $\mathrm{REF}_{v,w}$.

2. Let $H := (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$ be a graph with resource constraints and invertible resource extension functions.

   The inversion of H is defined to be the graph

   $$H^{-1} := \left(V_H, A_H^{-1}, \sqsupseteq, I, \mathrm{REF}^{-1}\right)$$

   with inverted arc set $A_H^{-1} := \left\{(w, v) \in V_H^2 \mid (v, w) \in A_H\right\}$ and inverted resource extension functions $\mathrm{REF}^{-1} := \left(\mathrm{REF}_{v,w}^{-1}\right)_{(w,v)\in A_H^{-1}}$.

Theorem 12 (Feasibility-conservation of inversions)**.** *Let $H := (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$ be a graph with resource constraints. Further, let all the resource extension functions of* REF *be invertible.*
Then a path $P := (v_0, \dots, v_n)$ of length $n \in \mathbb{N}_0$ in H is resource-feasible with witnessing resource vectors $(r^v)_{v\in P}$ if and only if $P^{-1} := (v_n, \dots, v_0)$ is a resource-feasible path in the inverted graph H^{-1} with witnessing resource vectors $(r^v)_{v\in P^{-1}}$.

This theorem is already proven in [Kai16, p. 83]. We need inversions in order to improve the behavior of the algorithm. Using inversions we can apply the algorithm once for all subproblems and do not have to solve each subproblem separately.

Label-Setting Algorithm

We solve the (SPPRC) via a label-setting algorithm. Previously in this section, we have added resources to the graph in order to restrict the set of feasible paths. Since we do not have a cost function, finding an optimal path is not straight-forward. We do not have a shortest path but multiple resource-feasible paths. We use the resource dominance relation $\sqsubseteq$ to compare different paths. If a path is dominated by another, it is not preferable and therefore not considered in the solution. If two paths are not comparable, we cannot decide which one is more preferable. This leads to the concept of Pareto-optimality.

Definition 25 (Pareto-optimal paths)**.** Let $H := (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$ be a graph with resource constraints. Let P be a resource-feasible v-w-path in H.

P is called Pareto-optimal if there exist witnesses $(r^u)_{u\in P}$ for the resource-feasibility of P such that for every resource-feasible v-w-path Q in H with witnessing resource vectors $(\tilde{r}^u)_{u\in Q}$ fulfilling $\tilde{r}^v = r^v$ holds:

$$r^w \sqsubseteq \tilde{r}^w \Rightarrow r^w = \tilde{r}^w$$

We say that the resource vectors $(r^u)_{u\in P}$ witness Pareto-optimality of P.

In general, there can be exponentially many paths in a graph with respect to its size. It is further possible no two of the witnessing resource vectors are comparable. Hence, there can be an exponential number of Pareto-optimal paths in a graph. This fact gives a feeling why (SPPRC) is $\mathcal{NP}$-hard.
The idea of the algorithm is to start with a trivial path, consisting of one single vertex. We then extend the paths while we maintain resource-feasibility and Pareto-optimality for all paths. Finally, we receive Pareto-optimal paths from the starting vertex to all vertices. The algorithm works on the concept of Dynamic Programming.

Algorithm 3: Label-setting algorithm for acyclic graphs with resource constraints

```
Input: graph with resource constraints $H := (V_H, A_H, \sqsubseteq, I, \mathrm{REF})$, topological
       sorting $v_0, \dots, v_n$ of $V_H$ and initial resource vector $r^{v_0}$
Output: shortest path tree rooted at $(v_0, r^{v_0})$ encoded by $\delta$
1  $\mathcal{P}_{v_0} \leftarrow \{r^{v_0}\}$ ;
2  $\delta(v_0, r^{v_0}) \leftarrow \emptyset$ ;
3  foreach $i \in \{1, \dots, n\}$ do $\mathcal{P}_{v_i} \leftarrow \emptyset$;
4  foreach $i = 0, \dots, n$ do
5      foreach $r^{v_i} \in \mathcal{P}_{v_i}$ do
6          foreach $w \in \mathrm{N}_H^+(v_i)$ do
7              $\mathcal{P} \leftarrow \max_{\sqsubseteq} \mathrm{REF}_{v_i,w}(r^{v_i})$ ;
8              foreach $r^w \in \mathcal{P}$ do $\delta(w, r^w) \leftarrow (v_i, r^{v_i})$;
9              $\mathcal{P}_w \leftarrow \mathcal{P}_w \cup \mathcal{P}$ ;
10             $\mathcal{P}_w \leftarrow \max_{\sqsubseteq} \mathcal{P}_w$ ;
11         end
12     end
13 end
14 return $\delta$
```

We have a graph with resource constraints, a topological sorting of the vertices and an initial resource vector as input. The graph has to be acyclic. A topological sorting $\{v_0, \dots, v_n\}$ means that there are no $i < j$ with $(v_j, v_i) \in A_H$. Starting with a vertex v_0 and an initial resource vector r^{v_0}, we treat the vertices successively in topological order. For each vertex $v \in V_H$ and for each computed Pareto-optimal v_0-v-path, we

try to extend the path feasibly by a single vertex. If a path is found, we add a label to the extending vertex and update the mapping δ in order to identify the origin of the extension. At the end, we receive the mapping δ which identifies all resource-feasible Pareto-optimal v_0-v-paths for each vertex $v \in V_H$.
In the following, we show how we apply the (SPPRC) and Algorithm 3 to the subproblem (SP_v).

5.2.2 Strengthening Inequalities

First we insert additional valid inequalities. They are not necessary but may improve the column generation process. In the original problem, we have the linking constraints (3.2) and (3.3) which ensure that for every customer exactly one route is fulfilled. These constraints cannot be moved into the subproblems since the customers can be satisfied by different vehicles. It is even likely that two trips of the same route are fulfilled by different vehicles.
Nevertheless, we can identify duties that are infeasible with respect to the cover constraints. This is the case, if a vehicle fulfills two trips that belong to the same customer but not to the same route. If this duty is part of the overall solution, it is not possible that (3.2) and (3.3) are fulfilled simultaneously. Therefore, we want to prevent such a duty from being added to the column set.
In order to use the inequalities, we have to introduce decision variables $u_m \in \{0, 1\}$ for $m \in \mathcal{M}$. We insert the following inequalities to (SP_v) for each $v \in \mathcal{V}$. They are not necessary since they are implied by the cover constraints of the master problem, but they strengthen the formulation of the subproblem.

$$\sum_{m \in C^{-1}(c)} u_m \leq 1 \qquad \text{for all } c \in \mathcal{C} \tag{5.9}$$

$$\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t} \leq u_m \qquad \text{for all } m \in \mathcal{M}, t \in m \tag{5.10}$$

These constraints ensure that for every customer at most one route can be fulfilled. Note that (5.9) is also valid with equality. Adding these constraints possibly makes the subproblems harder to solve, but improves the behavior in the master problem since there are no duties added that are infeasible from the very beginning. If the addition is beneficial for the overall process, is not known in advance.

5.2.3 Determination of the Resources

We define the resources that are needed to solve the (SP_v). The only resource that we have used so far is the fuel resource. We use the index *fuel* in our resource vector. The fuel is in the interval $[0, 1]$ for the fuel level where 0 means that the vehicle has

no fuel and 1 that the vehicle is completely fueled. A higher fuel level is preferable, hence the resource relation for fuel is $\leq$.
Since we have no objective function in this problem, we model the reduced cost as an additional resource. We introduce the index *redcost* in order to keep track of the reduced cost of a duty. The reduced cost is unrestricted and a smaller reduced cost is preferable, thus we set the resource window to $\mathbb{R}$ and the resource relation to $\geq$.
An additional resource is the number of trips that a vehicle fulfills. We call it the length of a duty and use the index *length*. This resource is not necessary for the subproblem but advantageous for the branch-and-bound procedure as we see in Section 5.3. The length of a duty lies in the interval $[0, |\mathcal{T}|]$. Comparing two duties with different lengths, it is not clear which of them is preferable. Thus the resource relation is $=$.
In order to ensure the constraints (5.9) and (5.10), we use a resource for every multimodal route. We use the respective $m \in \mathcal{M}$ as index for the route resource. This resource indicates how many trips of this route can still be fulfilled within this duty. Initially, all trips of a route can be fulfilled, thus the resource window is $[0, |m|]$ for $m \in \mathcal{M}$. Similar to the duty length it is not possible to compare different route resources, therefore the resource relation is $=$.
Altogether, we consider resources $\mathcal{U} := \{\text{redcost}, \text{fuel}, \text{length}\} \cup \mathcal{M}$ in the resource window

$$I := \mathbb{R} \times [0,1] \times [0, |\mathcal{T}|] \times \bigotimes_{m \in \mathcal{M}} [0, |m|] \subseteq \mathbb{R}^{\mathcal{U}}.$$

A resource vector $r \in I$ consists of the reduced cost $r_{\text{redcost}} \in \mathbb{R}$, the fuel level $r_{\text{fuel}} \in [0,1]$, the duty length $r_{\text{length}} \in [0, |\mathcal{T}|]$ and the route resources $r_m \in [0, |m|]$ for $m \in \mathcal{M}$ in this order. The resource relation vector is given by $\sqsubseteq := (\geq, \leq, =, =, \dots, =)$.
These resources coincide in large parts with the formulation in [Kai16]. They use resources for each customer instead of route resources in order to ensure the single-leg cover constraints. This is the only adaption of the resources.

5.2.4 Determination of the Resource Extension Function

In this section, we determine the resource extension function such that we receive a formulation equivalent to the (SP_v). We first define a more complex graph with simple resource extension functions. Then we contract the graph in order to shift the complexity from the graph into the functions.

Extended Task Graph with Split Vertices

In the (SSPRC) as defined in Section 5.2.1, changes of the resource vectors are only defined on the arcs. This means, only resources occurring on the arcs are considered. Since we have also a trip cost and fuel consumption at the vertices of the task graph, we

modify the graph in order to deal with vertex resources. Therefore, we create a graph by splitting the vertices, based on the extended task graph $\widehat{G} = \left(\widehat{V}, \widehat{A}\right)$ according to Definition 8.
We define the extended task graph with split vertices $\widetilde{G} := \left(\widetilde{V}, \widetilde{A}\right)$ with vertex set

$$\widetilde{V} := \{d^{\mathrm{s}}, d^{\mathrm{e}}\} \mathbin{\dot\cup} \left\{s^-, s^+ \mid s \in \widehat{V} \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}\right\}$$

and arc set

$$\begin{aligned}\widetilde{A} := &\left\{(d^{\mathrm{s}}, s^-) \mid (d^{\mathrm{s}}, s) \in \widehat{A}\right\} \mathbin{\dot\cup} \left\{\left(s^+, d^{\mathrm{e}}\right) \mid (s, d^{\mathrm{e}}) \in \widehat{A}\right\} \\ &\mathbin{\dot\cup} \left\{\left(s^+, t^-\right) \mid (s,t) \in \widehat{A}\right\} \mathbin{\dot\cup} \left\{\left(s^-, s^+\right) \mid s \in \widehat{V} \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}\right\}.\end{aligned}$$

Based on the graph $\widetilde{G}$, we define the resource extension functions $\widetilde{\mathrm{REF}}$. For this, we use $\sqsubseteq$ and I as described in Section 5.2.3. The values γ_t for $t \in \mathcal{T}$ and μ_v for the specific $v \in \mathcal{V}$ come from the dual solution of (DLRMP), from which the subproblem results. We define the resource extension function

$$\widetilde{\mathrm{REF}}_{t,t'}(c, e, l, b) \qquad\qquad \text{for } (t, t') \in \widetilde{A} \text{ and } (c, e, l, b) \in I.$$

The values c, e, l denote the reduced cost, fuel and length resource, respectively. The value $b = (b_m)_{m \in \mathcal{M}}$ stands for the route resources.
In the arc between the split vertices of a vehicle $v \in \mathcal{V}$ occurs the reduced cost $-\mu_v$ and the fuel consumption $(1 - f_v^0)$ due to the initial fuel f_v^0. Thus we define the resource extension function for $v \in \mathcal{V}$ as:

$$\widetilde{\mathrm{REF}}_{v^-,v^+}(c, e, l, b) := \left(\begin{pmatrix} c - \mu_v \\ e - (1 - f_v^0) \\ l \\ b \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0}\right) \cap I$$

In refuel points we have the fuel consumption f_r^{t}, thus we define the resource extension function for each refuel point r as:

$$\widetilde{\mathrm{REF}}_{r^-,r^+}(c, e, l, b) := \left(\begin{pmatrix} c \\ e - f_r^{\mathrm{t}} \\ l \\ b \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0}\right) \cap I$$

If a trip $t \in \mathcal{T}$ is fulfilled in this duty, no other trip of the same customer must be fulfilled, unless it belongs to the same route. Therefore, we introduce an auxiliary function. For $t \in \mathcal{T}$ and the route resource vector $b \in \bigotimes_{m\in\mathcal{M}} [0, |m|]$, we define

$$a^t : \bigotimes_{m\in\mathcal{M}} [0, |m|] \to \bigotimes_{m\in\mathcal{M}} [-1, |m|] \,,$$

$$a^t_m(b) = \begin{cases} b_m & \text{if } (C \circ M)(t) \neq C(m) \\ b_m - 1 & \text{if } M(t) = m \\ 0 & \text{if } (C \circ M)(t) = C(m) \text{ and } M(t) \neq m. \end{cases}$$

If trip $t \in \mathcal{T}$ is covered by the vehicle, then at most one trip less of the same route can be fulfilled. No other trip that belongs to another route of the same customer can be fulfilled, while the routes of other customers are not affected. If we have already $b_{M(t)} = 0$, then $a^t_{M(t)}(b) \cap [0, 1] = \emptyset$ and then t cannot be fulfilled.
In $t \in \mathcal{T}$ further occurs the cost $c^{\mathrm{t}}_t - \gamma_t$, the fuel consumption is f^{t}_t and the length of the duty increases by 1. Therefore, we define the resource extension function for $t \in \mathcal{T}$ as:

$$\widetilde{\mathrm{REF}}_{t^-,t^+}(c, e, l, b) := \left(\begin{pmatrix} c + c^{\mathrm{t}}_t - \gamma_t \\ e - f^{\mathrm{t}}_t \\ l + 1 \\ a^t(b) \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0} \right) \cap I \tag{5.11}$$

Between two trips occurs the cost and the fuel consumption for the deadhead trip. Therefore we have for $(s, t) \in A \cap (\mathcal{T} \cup \mathcal{R})^2$:

$$\widetilde{\mathrm{REF}}_{s^+,t^-}(c, e, l, b) := \left(\begin{pmatrix} c + c^{\mathrm{d}}_{s,t} \\ e - f^{\mathrm{d}}_{s,t} \\ l \\ b \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0} \right) \cap I$$

Between a vehicle and a trip we additionally have to consider the fixed vehicle cost c^{v}, thus the resource extension function for $(s, t) \in A \cap (\mathcal{V} \times (\mathcal{T} \cup \mathcal{R}))$ is defined as:

$$\widetilde{\mathrm{REF}}_{s^+,t^-}(c, e, l, b) := \left(\begin{pmatrix} c + c^{\mathrm{v}} + c^{\mathrm{d}}_{s,t} \\ e - f^{\mathrm{d}}_{s,t} \\ l \\ b \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0} \right) \cap I$$

For all arcs incident with d^{s} or d^{e}, the resource extension function is the function corresponding to the identity, i. e. $\left((c, e, l, b) + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0}\right) \cap I$.

Extended Task Graph

We transform the extended task graph with split vertices $\widetilde{G} = \left(\widetilde{V}, \widetilde{A}, \sqsubseteq, I, \widetilde{\mathrm{REF}}\right)$ by contracting the arcs $\left(t^-, t^+\right)$ for all $t \in \widehat{V} \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}$. Then we identify t^+ with t for all $t \in \widehat{V} \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}$ and receive the extended task graph $\widehat{G} = \left(\widehat{V}, \widehat{A}, \sqsubseteq, I, \widehat{\mathrm{REF}}\right)$ with

$$\widehat{\mathrm{REF}}_{s,t} = \widetilde{\mathrm{REF}}_{t^-,t^+} \circ \widetilde{\mathrm{REF}}_{s^+,t^-}$$

for all $s, t \in \widehat{V} \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}$. The graph $\widehat{G} = \left(\widehat{V}, \widehat{A}\right)$ is the extended task graph according to Definition 8. For arcs starting from a refuel point or a trip and leading to a trip, i. e. $(s, t) \in \widehat{A}, s \notin \mathcal{V}, t \in \mathcal{T}$, we have:

$$\widehat{\mathrm{REF}}_{s,t}(c, e, l, b) := \left(\begin{pmatrix} c + c^{\mathrm{d}}_{s,t} + c^{\mathrm{t}}_{s,t} - \gamma_t \\ e - f^{\mathrm{d}}_{s,t} - f^{\mathrm{d}}_{s,t} \\ l + 1 \\ a^t(b) \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsupseteq 0} \right) \cap I$$

since $f^{\mathrm{d}}_{s,t} \geq 0$ and $f^{\mathrm{t}}_{t} \geq 0$. The resources are independent from each other.
For arcs starting at a trip and leading to a refuel point, i. e. $(s, r), (s, t) \in A$, $s \in \mathcal{T}$, $r \in \mathcal{R}_{s,t}$, we have:

$$\widehat{\mathrm{REF}}_{s,r}(c, e, l, b) := \begin{cases} \left(\begin{pmatrix} c + c^{\mathrm{d}}_{s,r} \\ e - f^{\mathrm{d}}_{s,r} - f^{\mathrm{t}}_{r} \\ l \\ b \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsupseteq 0} \right) \cap I & \text{if } e \geq f^{\mathrm{d}}_{s,r} \\ \emptyset & \text{otherwise} \end{cases}$$

Note that $f^{\mathrm{t}}_{r} \leq 0$.
For both cases, we extend the evaluation to $s \in \mathcal{V}$ by adding c^{v} to the reduced cost resource. For arcs incident with d^{s} or d^{e} we have:

$$\widehat{\mathrm{REF}}_{d^{\mathrm{s}},s} = \widetilde{\mathrm{REF}}_{s^-,s^+} \qquad \widehat{\mathrm{REF}}_{s,d^{\mathrm{e}}} = \widetilde{\mathrm{REF}}_{s^+,d^{\mathrm{e}}} \qquad \text{for } s \in \mathcal{V} \cup \mathcal{T}$$

Task Graph

We transform the extended task graph $\widehat{G} = \left(\widehat{V}, \widehat{A}, \sqsubseteq, I, \widehat{\mathrm{REF}}\right)$ by contracting the arcs $(r, t) \in \widehat{A}$ for all $s, t \in \mathcal{V} \cup \mathcal{T}$ with $s \prec t$ and $r \in \mathcal{R}_{s,t}$. This yields the task graph $G = (V, A, \sqsubseteq, I, \mathrm{REF})$ which builds on the task graph $G = (V, A)$ according to Definition 7.

For every arc $(s,t) \in A$ with $s \notin \mathcal{V}, t \in \mathcal{T}$ and $r \in \mathcal{R}_{s,t}$, we determine the resource vectors $\left(\widehat{\mathrm{REF}}_{r,t} \circ \widehat{\mathrm{REF}}_{s,r}\right)(c,e,l,b)$ as

$$\begin{cases} \left(\begin{pmatrix} c + c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t} + c^{\mathrm{t}}_{t} - \gamma_t \\ \min\left(e - f^{\mathrm{d}}_{s,r} - f^{\mathrm{t}}_{r}, 1\right) - f^{\mathrm{d}}_{r,t} - f^{\mathrm{t}}_{t} \\ l+1 \\ a^t(b) \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0}\right) \cap I & \text{if } e \geq f^{\mathrm{d}}_{s,r} \\ \emptyset & \text{otherwise.} \end{cases}$$

For $s \in \mathcal{V}$, we add c^{v} to the reduced cost resource. According to Definition 23, the resource extension function is then given by

$$\mathrm{REF}_{s,t}(c,e,l,b) = \widehat{\mathrm{REF}}_{s,t}(c,e,l,b) \cup \bigcup_{r \in \mathcal{R}_{s,t}} \left(\widehat{\mathrm{REF}}_{r,t} \circ \widehat{\mathrm{REF}}_{s,r}\right)(c,e,l,b)$$

$$= \left[\left(\begin{pmatrix} c + c^{\mathrm{d}}_{s,t} + c^{\mathrm{t}}_{t} - \gamma_t \\ e - f^{\mathrm{d}}_{s,t} - f^{\mathrm{t}}_{t} \\ l+1 \\ a^t(b) \end{pmatrix} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0}\right) \cap I\right]$$

$$\cup \left[\left(\left\{\begin{pmatrix} c + c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t} + c^{\mathrm{t}}_{t} - \gamma_t \\ \min\left(e - f^{\mathrm{d}}_{s,r} - f^{\mathrm{t}}_{r}, 1\right) - f^{\mathrm{d}}_{r,t} - f^{\mathrm{t}}_{t} \\ l+1 \\ a^t(b) \end{pmatrix} \mid r \in \mathcal{R}_{s,t}, e \geq f^{\mathrm{d}}_{s,r}\right\} + \mathbb{R}^{\mathcal{U}}_{\sqsubseteq 0}\right) \cap I\right]$$

for $s \notin \mathcal{V}$ and with an additional c^{v} in the first component of all vectors for $s \in \mathcal{V}$.

5.2.5 Algorithms for Solving the Subproblems

After specification of the resources and the resource extension function, we present an algorithm that solves the subproblem for every vehicle optimally.

Model Equivalence

First, we prove that the previously developed problem formulation is equivalent to the subproblem and that the values for the reduced cost coincide.

Theorem 13 (Correspondence to (MMILP))**.** *For every resource-feasible d^{s}-d^{e}-path $P := (d^{\mathrm{s}}, v, t_1, \ldots, t_n, d^{\mathrm{e}})$ with $n \in \mathbb{N}_0$ in the graph $G = (V, A, \sqsubseteq, I, \mathrm{REF})$ with witnessing resource vectors $(r^v)_{v \in P}$, there is a feasible solution $(x,z,e,u) \in X_v$ satisfying (5.9) and (5.10) such that*

$$g^v(x,z,e,u) \leq r^{d^{\mathrm{e}}}_{\mathrm{redcost}} - r^{d^{\mathrm{s}}}_{\mathrm{redcost}}.$$

Inversely, for every feasible solution $(x, z, e, u) \in X_v$ *to some subproblem* $v \in \mathcal{V}$ *that satisfies* (5.9) *and* (5.10)*, there is a resource-feasible* d^{s}*-*d^{e}*-path* $P := (d^{\mathrm{s}}, v, t_1, \dots, t_n, d^{\mathrm{e}})$*,* $n \in \mathbb{N}_0$ *in* $G = (V, A, \sqsubseteq, I, \mathrm{REF})$ *with witnessing resource vectors* $(r^v)_{v\in P}$ *such that the equation* $g^v(x, z, e) = r^{d^{\mathrm{e}}}_{\mathrm{redcost}} - r^{d^{\mathrm{s}}}_{\mathrm{redcost}}$ *holds.*

This theorem is proven by [Kai16, pp. 96-99] for a slightly modified problem. The strengthening inequality is adapted to single-leg cover constraints and the route resource is a customer resource there. This has also an impact on the resource extension function. We do not present the complete proof, but only the parts that are different here. This concerns mainly the decision variable u_m and the route resources r_m for $m \in \mathcal{M}$.

Proof. We show that we can create a solution of (MMILP) out of a resource-feasible path and vice versa.

„⇒“: Resource-feasible path to feasible solution

Let $P := (d^{\mathrm{s}}, t_0, t_1, \dots, t_n, d^{\mathrm{e}}), n \in \mathbb{N}_0$ be a resource-feasible d^{s}-d^{e}-path in $G = (V, A, \sqsubseteq, I, \mathrm{REF})$ with witnessing resource vectors $(r^s)_{s\in P}$. Construct the following solution (x, z, e, u): For $a \in A$, set x_a to 1 if the path P uses the arc a and 0 otherwise. Set $e_s := r^s_{\mathrm{fuel}}$ if $s \in P$ and $e_s := 0$ otherwise. For all $i \in [n]$ and $r \in \mathcal{R}_{t_{i-1},t_i}$, set $z_{t_{i-1},r,t_i} := 1$ if $r^{t_{i-1}} \in \left(\mathrm{REF}_{r,t_i} \circ \mathrm{REF}_{t_{i-1},r}\right)\left(r^{t_{i-1}}\right)$ and $z_{t_{i-1},r,t_i} := 0$ otherwise. If this holds true for more than one $r \in \mathcal{R}_{t_{i-1},t_i}$, change z_{t_{i-1},r,t_i} to 0 for all such r but one. For all other arcs $a \in A$, set $z_a := 0$. For all $m \in \mathcal{M}$ set $u_m := 1$ if there is an $i \in [n]$ such that $M(t_i) = m$ and $u_m := 0$ otherwise. We claim that $(x, z, e, u) \in X_v$, i. e. (x, z, e, u) is a feasible solution to the subproblem of vehicle $v := t_0$.
The flow conservation and the fuel constraints hold as proven by [Kai16]. This applies directly since the respective parts of X_v and REF have not been modified. The claim concerning the reduced cost is also proven there.
Due to (3.4) and (3.11) we have $\sum_{s\in \mathrm{N}^-_G(t)} x_{s,t} \in \{0, 1\}$ for all $t \in \mathcal{T}$. From the construction follows that

$$\sum_{s\in \mathrm{N}^-_G(t)} x_{s,t} = 1 \qquad \Rightarrow \qquad t \in P \qquad \Rightarrow \qquad u_{M(t)} = 1$$

for all $t \in \mathcal{T}$ and therefore (5.10). For constraint (5.9), we assume by contradiction that there is a $m \in \mathcal{M}$ with $\sum_{m\in C^{-1}(c)} u_m > 1$. Then there exist $i, j \in [n]$ with $i < j$ such that

$$t_i, t_j \in P \qquad \text{and} \qquad M(t_i) \neq M(t_j) \qquad (M \circ C)(t_i) = (M \circ C)(t_j).$$

Due to the definition of $\mathrm{REF}_{t_{i-1},t_i}$ we have $a^{t_i}_{M(t_j)}(b) = 0$ for all $b \in I_{\mathcal{M}}$. The route resource is monotonously decreasing along each path. Therefore

$$r^{t_i}_{M(t_j)} = 0 \quad \Rightarrow \quad r^{t_{j-1}}_{M(t_j)} = 0 \quad \Rightarrow \quad r^{t_j}_{M(t_j)} = r^{t_{j-1}}_{M(t_j)} - 1 = -1 \quad \Rightarrow \quad r^{t_j} \notin I.$$

This leads to contradiction to the resource-feasibility of P and therefore holds (5.9).

„⇐“: Feasible solution to resource-feasible path

Let $(x, z, e, u) \in X_v$ for some $v \in \mathcal{V}$. Set $t_0 := v$. By the definition of X_v holds $x_{d^s,t_0} = 1$. Based on the flow conservation (3.4) and construction of the task graph G, the existence of exactly one $t_1 \in \mathcal{T} \cup \{d^e\}$ with $x_{t_0,t_1} = 1$ follows. This step can be repeated $n \in \mathbb{N}_0$ times for finding $t_i \in \mathcal{T} \cup \{d^e\}, i = 2, \dots, n+1$ until $t_{n+1} = d^e$ is reached. This defines a path $P := (d^s, t_0, \dots, t_n, d^e)$ in the task graph. Set the resource vectors $(r^s)_{s \in P}$ along this path as follows. For all vertices $t_k, k \in \{0, \dots, n\}$, set the reduced cost

$$r^{t_k}_{\text{redcost}} := c^{\text{v}} + \sum_{i=1}^{k} \Bigg[x_{t_{i-1},t_i} \left(c^{\text{d}}_{t_{i-1},t_i} + c^{\text{t}}_{t_i} - \gamma_{t_i} \right) + \sum_{r \in \mathcal{R}_{t_{i-1},t_i}} z_{t_{i-1},r,t_i} \left(c^{\text{d}}_{t_{i-1},r} + c^{\text{d}}_{r,t_i} - c^{\text{d}}_{t_{i-1},t_i} \right) \Bigg] - \mu_v,$$

the fuel level $r^{t_k}_{\text{fuel}} := e_{t_k}$, the length $r^{t_k}_{\text{length}} := k$ and the route resources

$$r^{t_k}_m = \begin{cases} 0 & \text{if} \displaystyle\sum_{\substack{i \in [k]:\\ (C \circ M)(t_i) = C(m)\\ M(t_i) \neq m}} \sum_{s \in \mathrm{N}^-_G(t_i)} x_{s,t_i} > 0 \\ |m| - \displaystyle\sum_{\substack{i \in [k]:\\ m = M(t_i)}} \sum_{s \in \mathrm{N}^-_G(t_i)} x_{s,t_i} & \text{otherwise} \end{cases}$$

for $m \in \mathcal{M}$. The route resource lies in the resource window since $|m|$ is chosen large enough. The resource vectors at the source and sink are set to $r^{d^s} := (0, 1, 0, (|m|)_{m \in \mathcal{M}})$ and $r^{d^e} := r^{t_n}$, respectively. We claim that these resource vectors $(r^s)_{s \in P}$ witness resource-feasibility of the path P in the task graph G.
The resource-feasibility for the resources *fuel*, *redcost* and *length* are already proven by [Kai16]. This applies directly since the respective parts of X_v and REF have not been modified.

For all $m \in \mathcal{M}$, we have $r_m^{t_0} = |m| \in I_m$. For $k \in [n]$, we distinguish the following cases for the resource-feasibility of $r_m^{t_k}$:

$$\left.\begin{array}{lll} m = M(t_k) & \Rightarrow & r_m^{t_k} = r_m^{t_{k-1}} - 1, r_m^{t_k} \geq 0 \\ C(m) \neq (C \circ M)(t_k) & \Rightarrow & r_m^{t_k} = r_m^{t_{k-1}} \\ \text{else} & \Rightarrow & r_m^{t_k} = 0 \end{array}\right\} \Rightarrow r_m^{t_k} \in \mathrm{REF}_{t_{k-1},t_k}\left(r_m^{t_{k-1}}\right)$$

We always have $r_m^{t_k} \geq 0$ since the initial value has been chosen large enough. Therefore, the above defined resource vector witnesses resource-feasibility for all $m \in \mathcal{M}$. This concludes the proof. □

Inversion of the Graph

We have seen in Theorem 13 that the subproblem (SP_v) can be written as a Shortest Path Problem with Resource Constraints. Therefore, we can apply Algorithm 3 in order to solve the subproblem optimally. Since there is a subproblem for every vehicle $v \in \mathcal{V}$, it is not advantageous to apply the algorithm for each subproblem separately. In order to improve this behavior we can exploit the symmetry of the problem. Remember, that the only differences of the subproblems are caused by the the dual variables (γ_t, μ_v) from the dual solution of (DLRMP). Thus, the subproblems vary only in the fact which vehicle is included. As we search d^{s}-d^{e}-paths in the subproblem, only the second vertex of such a path is affected.
From the results of Theorem 12 and Theorem 13 we can see that (SPPRC) given on the inverted task graph G^{-1} is equivalent to the one given by G. As Algorithm 3 yields Pareto-optimal paths from the starting vertex to each vertex in the graph, it is profitable to apply the algorithm to G^{-1} with starting vertex d^{e}. Thus, we receive Pareto-optimal d^{e}-v-paths for all $v \in \mathcal{V}$ in the inverted graph and we have to execute the algorithm only once. We can then extend the respective d^{e}-v-paths to d^{e}-d^{s}-paths and receive solutions for all the subproblems.
The following lemma shows that it is possible to invert the problem. It is a condition for Theorem 12 that the resource extension function is invertible.

Lemma 7 (Invertibility of REF)**.** *The resource extension function* REF *is invertible.*

Proof. We will only prove that the resource extension function $\widetilde{\mathrm{REF}}$ restricted to the route constraints and to the split vertices of trips is invertible. [Kai16] has already shown that the resource extension function for the simplified problem is invertible setting and that invertibility is maintained if arcs are contracted. Since the parts of $\widetilde{\mathrm{REF}}$ are independent from each other, the invertibility of REF follows directly.
Let $t \in \mathcal{T}$ be arbitrary. For readability, we write the restriction of the resource extension function to the route resources $\widetilde{\mathrm{REF}}_{t^-,t^+}(c, e, l, b)$ as $F_{t^-,t^+}(b)$. The restricted resource window is $I_{\mathcal{M}} := \bigotimes_{m \in \mathcal{M}} [0, |m|]$ and the resource dominance relation

is $\sqsubseteq_{\mathcal{M}} := (=, \dots, =)$. Further we define the set

$$\overline{M}(t) := C^{-1}\left((C \circ M)(t)\right) \setminus \{M(t)\}$$

as the routes that must not be fulfilled after t.
Due to the definition of $\widetilde{\mathrm{REF}}$, we have

$$F_{t^-,t^+}(b) = \left(a^t(b) + \mathbb{R}^{\mathcal{M}}_{\sqsubseteq_{\mathcal{M}} 0}\right) \cap I_{\mathcal{M}}.$$

for $t \in \mathcal{T}$ and $b \in I_{\mathcal{M}}$. Due to the definition of $\sqsubseteq_{\mathcal{M}}$, we can rewrite this expression as follows:

$$F_{t^-,t^+}(b) = \left\{a^t(b) \mid b_{M(t)} \geq 1\right\}$$

Further, we examine the preimage of the auxiliary function a^t. This is given by

$$\left(a^t_m\right)^{-1}(b) = \begin{cases} \{b_m + 1\} & \text{if } m = M(t) \\ [0, |m|] & \text{if } m \in \overline{M}(t) \\ \{b_m\} & \text{otherwise} \end{cases}$$

for $\left\{b \in I_{\mathcal{M}} \mid b_{M(t)} \leq |M(t)| - 1, b_m = 0 \text{ for all } m \in \overline{M}(t)\right\}$ and $\left(a^t\right)^{-1}(b) = \emptyset$ otherwise.
According to Definition 24, the inverted relation is given by

$$F^{-1}_{t^-,t^+} := \left\{\left(r^{t^+}, r^{t^-}\right) \mid \left(r^{t^-}, r^{t^+}\right) \in F_{t^-,t^+}\right\}$$

for all $t \in \mathcal{T}$. Applying the respective definitions we get:

$$\begin{aligned} F_{t^-,t^+} &= \left\{\left(b, a^t(b)\right) \quad \mid b \in I_{\mathcal{M}}, b_{M(t)} \geq 1\right\} \\ F^{-1}_{t^-,t^+} &= \left\{\left(a^t(b), b\right) \quad \mid b \in I_{\mathcal{M}}, b_{M(t)} \geq 1\right\} \\ &= \left\{ (b, b') \quad \mid b \in I_{\mathcal{M}}, b_{M(t)} \leq |M(t)| - 1, b_m = 0 \text{ for all } m \in \overline{M}(t), \right. \\ &\qquad \left. b' \in \bigotimes_{m \in \mathcal{M}} \left(a^t_m\right)^{-1}(b)\right\} \end{aligned}$$

We show that $F^{-1}_{t^-,t^+}$ is a resource extension function with respect to $\sqsupseteq_{\mathcal{M}}$ for all $t \in \mathcal{T}$. Since $\sqsubseteq_{\mathcal{M}} = (=, \dots, =)$ holds $(\sqsupseteq_{\mathcal{M}}) = (\sqsubseteq_{\mathcal{M}})$. Thus, for $r^v, \tilde{r}^v \in I_{\mathcal{M}}$ holds $r^v \sqsupseteq \tilde{r}^v \Leftrightarrow r^v = \tilde{r}^v$ and the conditions of Definition 20 and Definition 21 are fulfilled. Considering the other resources and contraction of vertices does not destroy the invertibility. Therefore REF is invertible. □

Heuristic Approach to the Subproblem

For the column generation approach, it is sufficient to receive a duty with negative reduced cost for each vehicle. This solution is not necessarily optimal w. r. t. the negative reduced cost. This follows directly from the formulation (5.8). Thus, we describe a heuristic approach for solving the subproblem. This is directly applied by [Kai16, pp. 104-107].
As the label-setting algorithm is very expensive as it create many labels, we try to decrease the number of possible labels by discretizing some continuous resources, e. g. the fuel resource. For this, we define a discrete set of allowed fuel levels as

$$\mathcal{E} \subseteq [0,1] \qquad 0,1 \in \mathcal{E} \qquad \text{and} \qquad \lfloor e \rfloor_{\mathcal{E}} := \max\{e' \in \mathcal{E} \mid e' \leq e\} \qquad \text{for } e \in [0,1].$$

Using the set of fuel levels $\mathcal{E}$, the fuel component of $\widehat{\text{REF}}_{s,t}$ for $s \in \mathcal{V} \cup \mathcal{T}$, $t \in \mathcal{T}$ changes to

$$e \mapsto \left(\left\lfloor e - f^{\text{d}}_{s,t} - f^{\text{t}}_{t} \right\rfloor_{\mathcal{E}}\right) \cap I_{\text{fuel}}$$

for $e \in I_{\text{fuel}} = [0,1]$. Similarly, the fuel component for $\widehat{\text{REF}}_{r,t} \circ \widehat{\text{REF}}_{s,r}$ reads as

$$e \mapsto \begin{cases} \left(\left\lfloor \min\left(e - f^{\text{d}}_{s,r} - f^{\text{t}}_{r}, 1\right) - f^{\text{d}}_{r,t} - f^{\text{t}}_{t} \right\rfloor_{\mathcal{E}} + \mathbb{R}_{\leq 0}\right) \cap I_{\text{fuel}} & \text{if } e \geq f^{\text{d}}_{s,r} \\ \emptyset & \text{otherwise.} \end{cases}$$

The modified REF with fuel levels $\mathcal{E}$ is also an invertible resource extension function. Further, each duty that is resource-feasible in the heuristic is also feasible in the original formulation. This is proven by [Kai16].
In the column generation process, we create columns with the heuristic methods. If the heuristic finds duties with negative reduced cost, we insert these duties to the index set. Only if we cannot find suitable duties we apply the optimal label-setting algorithm. If the optimal algorithm does not find any duties with negative reduced cost any more, we have reached an optimal solution to the relaxed master problem.

5.3 Solving the Master Problem

With the methods developed in Section 5.1 and Section 5.2 we are able to solve the relaxed master problem. Having a solution of this relaxed problem, it is not guaranteed that the solution is an integer solution. However, this is a condition for feasibility in the master problem. If the solution is a fractional solution, we apply a „Branch-and-Bound“ process in order to receive an integer solution. We describe how Branch-and-Bound works in general and how we can apply this to our problem. Then we discuss a number of possible branching rules. Finally, we show how the branching rules are chosen. The branching rules are mainly taken over by the branching rules created by [Kai16]. The branching rule concerning the choice of routes is modified.

5.3.1 Branch-and-Bound

The set of feasible solutions for the relaxed problem is divided into two separate sets by a hyperplane. We express this restriction as an inequality. We evaluate both sets individually. An optimal solution of one of these sets is an optimal solution for the entire problem. If we repeat this procedure iteratively, we get smaller problems. This procedure is called „Branching“. We receive a tree of smaller problems where the root is the original master problem and the respective child nodes are the smaller problems resulting by branching.

The overall process works as follows: Starting with the root, we solve the respective relaxed problem. If we receive a fractional solution, we branch the problem and create child nodes. We continue this procedure for each child until we either receive a feasible integer solution or the problem becomes infeasible. After this, we iteratively assign to each node the feasible solution of its child nodes with the smallest objective value, or infeasible, if both child nodes are infeasible. Therefore, the solution assigned to the root is the optimal solution of the entire problem.

Depending on the problem and the branching strategy, the decision tree might become quite large. Therefore we have to think about methods to improve this behavior. If we already have an initial feasible solution, then the value of this solution is an upper bound to the optimal solution value. As we know, the value of the relaxation is a lower bound to the optimal value of this problem. Therefore, if the value of the relaxation for a subtree is greater than the value of the initial solution, we know that the optimal solution does not lie in this subtree. Thus we can completely neglect this subtree. This method is called „Bounding“.

Application to the Master Problem

A branching decision is always an inequality that we add to the master problem. As suggested by [Kai16], we only use inequalities written in terms of $(x, z, e, u) \in X_v, v \in \mathcal{V}$ such that the structure of the subproblems is not changed. Further, we do not want to affect the symmetry of the subproblems since we exploit this when we solve the subproblems. Thus, we discuss the decision rules with respect to keeping the symmetry.

If an inequality concerns only one single subproblem, we move this restriction completely to the subproblem. This has the advantage that we still create feasible columns. However, moving the inequalities to the subproblems might lead to a change of their structure and therefore might make it harder to solve. In Section 5.3.2 we discuss several branching rules.

5.3.2 Branching Rules

In the following, we present a number of branching rules. In order to keep the branching tree small, we try to create decisions that lead to a balanced branching. The branching is used for enforcing integrality in the master problem. In order to discuss the branching rules, we introduce for each vertex $v \in V$ the set of vertices that can be reached from v and the set of vertices from which v can be reached:

$$\mathrm{N}_G^{++}(v) := \{w \in V \mid \exists v\text{-}w\text{-path in } G\} \qquad \mathrm{N}_G^{--}(v) := \{w \in V \mid \exists w\text{-}v\text{-path in } G\}$$

Assignment of Trips

We first consider a branching on the components of the image with respect to ψ. This is a branching on the single variables of the master problem and thus the most specific branching rule we consider. We fix a value for the expression

$$\psi\left(x^v, z^v, e^v\right)_t = \sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v \in \{0,1\}$$

for some $v \in \mathcal{V}, t \in \mathcal{T}$. This decision can be interpreted as determining whether trip t is fulfilled by vehicle v. This is a very specific decision since it implicitly chooses the route $M(t)$ for customer $(M \circ C)(t)$ and assigns trip t to vehicle v.
Branching down means setting $\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v = 0$ for some $v \in \mathcal{V}, t \in \mathcal{T}$. This can be implemented in (SP_v) by setting

$$x_{s,t}^v = 0 \qquad\qquad \text{for all } s \in \mathrm{N}_G^-(t).$$

This corresponds to deleting the respective arcs in the task graph corresponding to subproblem v.
Branching up means demanding $\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v = 1$ for some $v \in \mathcal{V}, t \in \mathcal{T}$. With the constraints (3.4), (5.1), (5.2) and the fact that G is acyclic, this is equivalent to setting

$$x_{s,u}^v = 0 \qquad \text{for all } (s,u) \in A \cap \left(\left(\mathrm{N}_G^{--}(t) \backslash \{t\}\right) \times \left(\mathrm{N}_G^{++}(t) \backslash \{t\}\right)\right).$$

This corresponds to deleting all arcs skipping trip t.
According to the cover constraints (3.3) and (5.3), branching up implicitly choose the route for customer $(C \circ M)(t)$. Therefore, it further demands

$$x_{s,t'}^v = 0 \qquad\qquad \text{for all } v \in \mathcal{V}, t' \in \overline{M}(t), s \in \mathrm{N}_G^-(t').$$

Hence all trips belonging to another route of the same customer are deleted. This applies for all subproblems simultaneously.

While branching up excludes a lot of possible assignments, branching down only forbids one such assignment. This leads to a quite unbalanced tree. Another disadvantage is that this rule destroys symmetry between the subproblems as it concerns only one subproblem. Nevertheless, there is always a branching decision that can be made if the previous solution was not integral. This means, this rule is enough to completely ensure integrality in the master problem.

Length of Vehicle Duties

Another suggestion is to consider not a single trip, but the sum up over all trips for the components of the image of ψ. We fix a value for the expression

$$\sum_{t \in \mathcal{T}} \psi\left(x^v, z^v, e^v\right)_t = \sum_{t \in \mathcal{T}} \sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v \in \{0, \ldots, |\mathcal{T}|\}$$

for some $v \in \mathcal{V}$. This can be interpreted as determining the duty length for vehicle v. As the length of the duty can be expressed linearly in terms of (x^v, z^v, e^v) and in terms of images of ψ it would be possible to include this decision in the master problem without changing the subproblems. But since this decision affects only one specific vehicle, we move it to the subproblem.
This leads to the following general setting of the subproblem using a lower bound l^{LB} and an upper bound l^{UB} for the vehicle length.

$$\begin{aligned} \min \quad & g^v\left(x^v, z^v, e^v\right) \\ \text{s.t.} \quad & l^{\mathrm{LB}} \leq \sum_{t \in \mathcal{T}} \sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v \leq l^{\mathrm{UB}} \\ & (x^v, z^v, e^v) \in X_v \end{aligned} \tag{5.12}$$

As proven by [Kai16], this problem can be solved by using the resource length as introduced before. We modify the resource extension function for $(c, e, l, b) \in I$ and $t \in \mathcal{V} \mathbin{\dot\cup} \mathcal{T}$ as follows:

$$\mathrm{REF}_{t,d^{\mathrm{e}}}(c, e, l, b) := \left(\left\{(c, e, l, b) \mid l^{\mathrm{LB}} \leq l \leq l^{\mathrm{UB}}\right\} + \mathbb{R}_{\sqsubseteq 0}^{\mathcal{U}}\right) \cap I$$

This is a coarser branching rule than the assignment of trips, but still destroys the symmetry between the subproblems. In contrast to before, this rule alone is not sufficient to completely ensure the integrality in the master problem.

Choice of Multimodal Routes

The next approach is to sum up over all vehicles for the components of the image of ψ. We fix a value for the expression

$$\sum_{v \in \mathcal{V}} \psi\left(x^v, z^v, e^v\right)_t = \sum_{v \in \mathcal{V}} \sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v \in \{0,1\}$$

for some $t \in \mathcal{T}$. This can be interpreted as deciding whether trip t is fulfilled by some vehicle. Because of the cover constraint (5.3), this decision directly determines the route variable u_m for $m := M(t)$ and therefore all other trips $t' \in M^{-1}(m)$. Because of this implication, we directly regard the case of branching on u_m.
Branching down means setting $u_m = 0$ for some $m \in \mathcal{M}$. By (5.3) follows $\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v = 0$ for all $t \in M^{-1}(m), v \in \mathcal{V}$ and therefore

$$x_{s,t}^v = 0 \qquad\qquad \text{for all } v \in \mathcal{V}, t \in M^{-1}(m), s \in \mathrm{N}_G^-(t).$$

This corresponds to deleting the respective arcs in all task graphs.
Branching up means setting $u_m = 1$ for some $m \in \mathcal{M}$ and therefore demanding $\sum_{s \in \mathrm{N}_G^-(t)} x_{s,t}^v = 1$ for all $t \in M^{-1}(m), v \in \mathcal{V}$. Due to the cover constraint (3.2) this is equivalent to setting $u_{m'} = 0$ for all $m' \in (C^{-1} \circ C)(m) \backslash \{m\}$. We therefore set

$$\begin{aligned} x_{s,t}^v = 0 \qquad\qquad & \text{for all } v \in \mathcal{V}, m' \in \left(C^{-1} \circ C\right)(m) \backslash \{m\}, \\ & \qquad t \in M^{-1}(m'), s \in \mathrm{N}_G^-(t). \end{aligned}$$

In other words, stating that a multimodal route is fulfilled by some vehicle is equivalent to stating that all trips belonging to other routes of the same customer are not fulfilled by any vehicle, if (3.2) and (5.3) hold. Again, this can be realized by deleting the respective arcs in all task graphs.
This branching rule maintains the symmetry for the various subproblems since for all $v \in \mathcal{V}$ the same arcs are deleted. Therefore, we can still solve all subproblems by just one execution of the algorithm. Further, the tree is more balanced than before. Similar to the duty length, this rule alone is not sufficient to completely ensure integrality in the master problem.

Number of Used Vehicles

Finally, we regard the branching on the number of used vehicles. We say, a vehicle is used if it fulfills at least one trip. The only duty that a vehicle can have to be unused is the duty with no trip, uniquely described by the path $(d^{\mathrm{s}}, v, d^{\mathrm{e}})$ in G. For a duty $(x, z, e) \in X_v, v \in \mathcal{V}$ the term

$$\sum_{(s,t)\in A\cap(\mathcal{V}\times\mathcal{T})} x^v_{s,t} \in \{0,1\}$$

is one if and only if the vehicles serves at least one trip. One possibility would be to decide if a specific vehicle is used or not. But the various subproblems are too similar as only the vehicle vertex is different and therefore this branching rule leads to an unbalanced tree.
Instead, we regard the number of used vehicles. For $(x^v, e^v, z^v) \in X_v$ the number of used vehicles can be expressed as

$$\sum_{v\in\mathcal{V}} \sum_{(s,t)\in A\cap(\mathcal{V}\times\mathcal{T})} x^v_{s,t} \in \{0, \dots, |\mathcal{V}|\}\,.$$

Using this branching decision requires adding a suitable inequality to the master problem. As these inequalities concern the duties of all vehicles, they cannot be moved to the subproblems. We modify the master problem using a lower bound v^{LB} and an upper bound v^{UB} for the number of used cars as follows:

$$\begin{aligned}
\min \quad & \sum_{v\in\mathcal{V}} g\left(x^v, z^v, e^v\right) + \sum_{m\in\mathcal{M}} u_m c^{\mathrm{r}}_m & & \\
\text{s.t.} \quad & \sum_{m\in C^{-1}(c)} u_m = 1 & \text{for all } c \in \mathcal{C} & \quad (3.2) \\
& \sum_{v\in\mathcal{V}} \sum_{s\in \mathrm{N}^-_G(t)} x^v_{s,t} = u_m & \text{for all } m \in \mathcal{M}, t \in m & \quad (5.3) \\
& v^{\mathrm{LB}} \le \sum_{v\in\mathcal{V}} \sum_{(s,t)\in A\cap(\mathcal{V}\times\mathcal{T})} x^v_{s,t} \le v^{\mathrm{UB}} & & \quad (5.13) \\
& (x^v, z^v, e^v) \in X_v & \text{for all } v \in \mathcal{V} & \\
& u_m \in \{0,1\}^{\mathcal{M}} & &
\end{aligned}$$

5.3.3 Choosing Branching Decisions

Consider, we have received a non-integer solution after solving the relaxed master problem. Then we need to choose a branching rule for the Branch-and-Bound process. Further we select the expression on which the branching rule is applied, e. g. $(v, t) \in \mathcal{V} \times \mathcal{T}$ for applying the rule for trip assignment. Using this, we create inequalities which define the resulting subproblems. The branching rule is only applicable, if the solution violates the inequalities. As mentioned before, in each non-integer solution the rule for trip assignment is applicable for some $(v, t) \in \mathcal{V} \times \mathcal{T}$.

We select the branching rule in a lexicographic order. We prefer coarser rules and rules that maintain the symmetry of the resulting tree. If applicable, we always branch on the number of vehicles. If not, we branch on the multimodal routes, the length of the duties and the assignment of trips, when one of the rules is applicable. After choosing the branching rule, we apply this on the variable which differs most from the nearest integer. Algorithm 4 describes the procedure.

Algorithm 4: Selection of a branching decisions

```
   Input: solution $(y, u)$ of (LMP) with $(y^v) \in \operatorname{conv}(\psi(X_v))$ for $v \in \mathcal{V}$ and
          $(u_m)_{m \in \mathcal{M}}$
   Output: branching inequalities or statement that solution is integer
1  $v_{\text{sum}} \leftarrow \sum_{v \in \mathcal{V}} (1 - \prod_{t \in \mathcal{T}} y_t^v)$;
2  if $v_{\text{sum}} \notin \mathbb{N}_0$ then
3      $v^{\text{UB}} \leftarrow \lfloor v_{\text{sum}} \rfloor$ and $v^{\text{LB}} \leftarrow \lceil v_{\text{sum}} \rceil$;
4  else if $\exists m \in \arg\min_{m' \in \mathcal{M}} |u_{m'} - 0.5|$ with $u_m \notin \mathbb{N}_0$ then
5      $u_m = 0$ and $u_m = 1$;
6  else if $\exists v \in \arg\min_{v' \in \mathcal{V}} |\mathbb{1}_{\mathcal{T}}^T y^{v'} - \lfloor \mathbb{1}_{\mathcal{T}}^T y^{v'} + 0.5 \rfloor|$ with $\mathbb{1}_{\mathcal{T}}^T y^v \notin \mathbb{N}_0$ then
7      $l^{\text{UB}} \leftarrow \lfloor \mathbb{1}_{\mathcal{T}}^T y^v \rfloor$ and $l^{\text{LB}} \leftarrow \lceil \mathbb{1}_{\mathcal{T}}^T y^v \rceil$;
8  else if $\exists (v, t) \in \arg\min_{(v', t') \in \mathcal{V} \times \mathcal{T}} |y_{t'}^{v'} - 0.5|$ with $y_t^v \notin \mathbb{N}_0$ then
9      $y_t^v = 0$ and $y_t^v = 1$;
10 else
11     Solution is integer;
12 end
```

The choice of the branching rules is based on the observations of [Kai16, Sec. 8.3]. They investigated the symmetry of the Branch-and-Bound tree for the respective branching rules and came up with this strategy.

Chapter 6

Instance Creation

In the following we explain, how the test instances are created. This concerns the sets of customers, routes, trips, vehicles and refuel points. Also the computation of the distances, times, costs and fuel states is discussed here.

6.1 Available Data

As the mathematical content of this thesis is based on two previous theses [Kai16] and [Kno16], we also use their data in order to test our solution approaches. There are two reasons that we cannot apply these test data directly. Firstly, because of the simplified problem setting, the respective data fulfill these restrictions. We need customers and routes that consist of more than one car trip in order to test all the abilities of the developed solution methods. Secondly, only exemplary instances that are strongly restricted in time are provided to us. Therefore we need to extract these exemplary instances in order to receive a full test instance.
The available test data are historical data that have been provided by the car sharing supplier Car2Go. The refuel point data have been provided by the electricity provider EnBW. The location of the data is Stuttgart. This suits best to the problem setting, since these data haven been recorded with electrically powered vehicles. The problem setting is constructed especially for the needs of electrically powered vehicles since then we have a restricted fuel range for the vehicles and a restricted number of available refuel points.
The trip set $\mathcal{T}$ of the provided test instance consists of 176 trips that all start on the same day in 2015 between 7:00 pm and 8:00 pm. The vehicle set $\mathcal{V}$ consists of 50 vehicles and the refuel point set $\mathcal{R}$ consists of 313 refuel points.

6.2 Trips, Vehicles and Refuel Points

We extend the available trip data to a full instance of 24 hours, where we aim a realistic distribution of the trips over time. We first create customers with associated start and end locations and a start time. Then we compute alternative routes for each of the customers.

Customer Creation

We are given a set of trips $\widehat{\mathcal{T}}$ with a start and end position and time for each of these trips. $\widehat{\mathcal{T}}$ is the trip set of the exemplary instance. In order to extend this trip set to a full day instance, we need a realistic distribution of the trips over time. In the previous theses, a distribution of the trips is provided. This is shown in Figure 6.1. For the trips of a whole month, we can see the maximum, average and minimum number of trips starting at this time point. As we can see, the time with the smallest number of starting trips is about 3 am. Therefore we assume that our full instance starts and ends at 3:00 am.

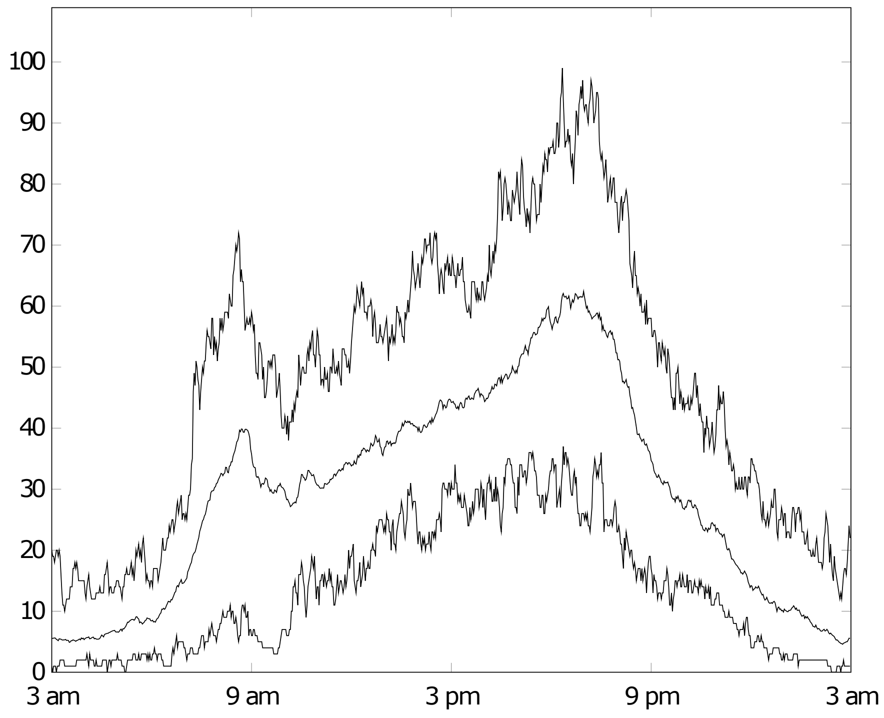

Figure 6.1: Distribution of trips ([Kai16, p. 62], [Kno16, p. 62])

We aim to receive customers with several alternative routes. All routes of the same customer lie in a similar time window. Therefore we first create a set of customers. For each customer we determine a start and end position and a start time. The distribution of the customers' start times is similar to the distribution of the trips in Figure 6.1. We do not make the customers to be simple copies of the given trip set over a full instance. Instead we set the start and end positions randomly. We split the full instances into time periods of one hour, starting at each full hour. For each time period we create customers as follows:

1. Determine the average number k of trips starting in this time period by the given distribution.
2. Take a sample of k trips out of $\widehat{\mathcal{T}}$, use their start positions and times for the customers.
3. Use the end positions of another sample of k trips as end positions for the customers.
4. Shift all start times by a full hour such that they start in the respective time period.

With this procedure we create the customer set $\mathcal{C}$ with a preliminary start and end position and start time for each customer. Note that in general this preliminary start time does not coincide with z_c^{start} which is used in the mathematical models.

Route Creation

For each customer we need a set of multimodal routes. Each route roughly starts at the preliminary start time of its customer. It further contains at least one car trip. For each customer there are routes with more than one car trip. In reality, such a route comprises a car trip, a public transport trip and then another car trip. Optionally, there are walking trips in-between. This is the only realistic possibility to gain routes with more than one car trip. Neither two car trips in a row nor a car trip between two public transport trips are a desirable option. The existence of routes with more than one car trip is crucial in order to test the developed solution approaches completely.
The multimodal routes are created with the open source software OpenTripPlanner (cf. [OTP]). The input for this software is a start and end position and a start time. For each input, it provides several alternative multimodal routes, that comprise car, public transport and walking trips. A great restriction of these route is that they contain only one car trip and this car trip is always at the beginning. In order to receive the required test instances, we have to think of appending a car trip to the existing routes in a realistic way. For this, we do a reverse search from the end point to the start point and identify the station at which the public transport trip starts. Then we search for a multimodal route to this station and append a car trip from this station to the end position. This gives us a multimodal route that starts and ends with a car trip.
For each customer, we are given a start and end position and a start time. We create the routes as follows:

1. Request a reverse search from the end position to the start position. Restrict the search to 3 alternative routes. For each alternative, identify the station where the public transport trip starts.

2. For each station, request a search from the start position to the station. Restrict the search to one alternative. Append a direct car trip from the station to the end position.

3. Request a search from the start position to the station. Restrict the search to 3 alternatives.

4. Create a route with a direct car trip from the start to the end position

We use the customer start time as the start time for each search request. From each route, we extract the car trips. They are the only trips that are part of the problem input. With this procedure we create the multimodal route set $\mathcal{M}$ and the trip set $\mathcal{T}$ with up to 7 alternative routes and 10 trips for each customer.

Vehicles and Refuel Points

Besides trips, a test instance contains vehicles and refuel points. Depending on the requested vehicle number, we take a sample of the previous vehicle set. If this does not suffice, i. e. the requested vehicle number is greater than the vehicle set, we additionally create vehicles with the trips' end positions as starting positions of the vehicles. Since we do not have further information about the availability of the vehicles, we choose 2:00 am as the starting time of all vehicles. For the refuel point set, we use a sample of the existing refuel point set with the requested size.

6.3 Fuel and Cost

We need to determine the distances and times between two points for the fuel states and the costs. We compute these values with the open source software OpenSource-RoutingMachine (cf. [OSR]). The input are start and end positions and it provides the time and distance between each pair of start and end positions. We request the distances d and the times t for

$$\begin{array}{lll} d_{s,t} & t_{s,t} & \text{for all } s \in \mathcal{V} \mathbin{\dot\cup} \mathcal{T} \mathbin{\dot\cup} \mathcal{R}, t \in \mathcal{T} \mathbin{\dot\cup} \mathcal{R} \\ d_t & t_t & \text{for all } t \in \mathcal{T} \end{array}$$

and set the end times $z_t^{\text{end}} := z_t^{\text{start}} + t_t$ for $t \in \mathcal{T}$.

Fuel States

We assume that the fuel consumption depends linearly on the driven distance of the electrically powered vehicles and the refuel rate depends linearly on the time the vehicle stays at the refuel point. The fuel state of a vehicle is always in the interval $[0, 1]$.

We introduce the constants v^{range} which describes the maximal range of a vehicle and v^{refuel} which describes the maximal refueling time. Therefore the fuel consumption is given by

$$f^{\text{d}}_{s,t} := \frac{d_{s,t}}{v^{\text{range}}} \quad \text{for } s \in \mathcal{V} \cup \mathcal{R} \cup \mathcal{T}, t \in \mathcal{R} \cup \mathcal{T} \qquad f^{\text{t}}_{t} := \frac{d_t}{v^{\text{range}}} \quad \text{for } t \in \mathcal{T}.$$

The refuel rate is fiven by

$$f^{\text{t}}_{r} := -\frac{z^{\text{start}}_t - z^{\text{end}}_s - t_{s,r} - t_{r,t}}{v^{\text{refuel}}} \qquad \text{for all } r \in \mathcal{R}_{s,t}.$$

Since we do not have further information about the fuel states of the vehicles, we set

$$f^{0}_{v} := 1 \qquad \text{for all } v \in \mathcal{V}.$$

As realistic values we assume that the vehicle range is 135km and the refueling time is 8h. Therefore we set $v^{\text{range}} = 135{,}000$ and $v^{\text{refuel}} = 28{,}800$.

Costs

The costs also depend linearly on the driven distance. The constant c^{meter} describes the cost for each meter a vehicle drives. Thus we set

$$c^{\text{d}}_{s,t} := c^{\text{meter}} \cdot d_{s,t} \quad \text{for } s \in \mathcal{V} \cup \mathcal{R} \cup \mathcal{T}, t \in \mathcal{R} \cup \mathcal{T} \qquad c^{\text{t}}_{t} := c^{\text{meter}} \cdot d_t \quad \text{for } t \in \mathcal{T}.$$

We set the constants $c^{\text{meter}} = 1$ and $c^{\text{v}} = 50{,}000$.
The only cost left is the route cost. As mentioned in Section 2.2, we use the route cost in order to model a realistic customer behavior. It contains the cost for public transport and the user inconvenience that arises if the customer chooses this route. We approximate the public transport cost in comparison to cost for car sharing. The constant c^{public} describes the cost for using public transport. The main user inconveniences are the total travel time and the walking time. The constants c^{time} and c^{walk} describe the respective penalty costs.
Based on the charges of Car2Go (cf. [Car]), we assume that a customer pays in average 0.50 euros per kilometer driving with a car. Due to the public transport provider in Stuttgart (cf. [VVS]), we assume that a customer pays in average 6 euros per hour using public transport. In order to compare the user inconveniences, we additionally charge 10 euros per hour for the total travel time and 10 euros per hour walking.

Example

In order to illustrate the procedure that we have developed before, we show an example of a multimodal route.

Time	Mode	d_t	c^{t}	c^{r}	Cost
8:00 - 8:12	Car	6km	6000		3.00 euros
8:12 - 8:15	Walk			1000	0.50 euros
8:15 - 8:25	Subway			2000	1.00 euros
8:25 - 8:28	Wait				
8:28 - 8:33	Bus			1000	0.50 euros
8:33 - 8:39	Walk			2000	1.00 euros
8:39 - 8:45	Car	3km	3000		1.50 euros

Table 6.1: Route according to Example 6

Example 6. Consider that the route as described in Table 6.1 has been developed in Section 6.2.
Only the car trips of this route are part of the input. Thus we have $t_1, t_2 \in \mathcal{T}$. We call the route $m = (t_1, t_2)$ and the associated customer $c \in \mathcal{C}$. The trips have the values:

$$\begin{aligned} z_{t_1}^{\text{start}} &= \text{8:00pm} & z_{t_1}^{\text{end}} &= \text{8:12pm} & c_{t_1}^{\mathrm{t}} &= 6000 & f_{t_1}^{\mathrm{t}} &= 0.044 \\ z_{t_2}^{\text{start}} &= \text{8:39pm} & z_{t_2}^{\text{end}} &= \text{8:45pm} & c_{t_2}^{\mathrm{t}} &= 3000 & f_{t_2}^{\mathrm{t}} &= 0.022 \end{aligned}$$

The route cost comprises a penalty of 15,000 for the total travel time and in summary we have $c_m^{\mathrm{r}} = 21{,}000$.

Chapter 7

Implementation and Results

In this chapter, we describe the implementation of the algorithms that we have developed in Chapter 4 and Chapter 5. All heuristical solution methods, based on the customer-dependent splitting and the time-dependent splitting and the iterative approach have been implemented and tested. Further, the optimal approach has been implemented and tested. The underlying test instances have been created as described in Chapter 6. Additionally, we provide computational results for all these algorithms and compare them with respect to computation time and results. Finally, we contextualize the results in the problem description as stated in Chapter 2.

7.1 Implementation

Similar to the mathematical models, we adapt great parts of the implementation from the underlying these [Kai16] and [Kno16]. Concerning the heuristics, on the one hand we limit ourselves to only one approach of the heuristics developed by [Kno16]. On the other hand, we extend heavily extend this approach into two different splitting methods and the iterative approach. In order to provide these methods and to keep the code clearly arranged, we modify the majority of the existing code for the heuristics. In contrast, the optimal approach developed by [Kai16] is quite similar. Thus, we reuse most of the code for the optimal approach and adjust it on a few positions.

Implementation Details

Before describing the implementation, we going into some details in the models and the algorithms that we have not yet stated in detail or that we slightly modified in the implementation.
We need to determine the number of partial instances n and the time points c_i, $i \in [n-1]$. An approach is to choose the time points iteratively, „such that a partial instance, that is as large as possible but still solvable to optimality in reasonable time, arises.“ ([Kno16, p. 131]) We do not examine, how large the partial instances may be. Instead, we determine all time points in advance such that the number of vehicles and trip is the same in each partial instance.

As opposed to Chapter 4, we do not consider all refuel points $r \in \mathcal{R}_{s,t}$ for some arc $(s,t) \in A$. Instead, we only consider one refuel point $r \in \mathcal{R}_{s,t}$ for which $c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t}$ is minimal, provided that $\mathcal{R}_{s,t} \neq \emptyset$. This involves a strong simplification in implementing the heuristic, as the task graph contains only one arc with additional information. Further it reduces the computation time due to much fewer possibilities. In the optimal approach, we use a set of Pareto-optimal refuel points w. r. t. a suitable function. This approach is examined by [Kai16, Sec. 3.2.2] and [Kno16, Sec. 3.2.2] and guarantees an optimal solution.
In the optimal approach, we model the cover constraints by using auxiliary variables u_m for $m \in \mathcal{M}$.

$$\sum_{m \in C^{-1}(c)} u_m = 1 \qquad \text{for all } c \in \mathcal{C} \tag{3.2}$$

$$\sum_{v \in \mathcal{V}} y^v_t = u_{M(t)} \qquad \text{for all } t \in \mathcal{T} \tag{5.4}$$

However, we can find an equivalent formulation without these auxiliary variables. For this, we define a function $T : \mathcal{M} \to \mathcal{T}$ with $(T \circ M)(m) = m$. This means, T maps a route to a fixed trip of the route, e. g. to its first trip. Then, we have $u_m = 1$ if and only if trip $T(m)$ is fulfilled and an equivalent formulation of the cover constraints reads as:

$$\sum_{m \in C^{-1}(c)} \sum_{v \in \mathcal{V}} y^v_{T(m)} = 1 \qquad \text{for all } c \in \mathcal{C}$$

$$\sum_{v \in \mathcal{V}} y^v_t = \sum_{v \in \mathcal{V}} y^v_{(M \circ T)(t)} \qquad \text{for all } t \in \mathcal{T} \setminus \bigcup_{m \in \mathcal{M}} T(m)$$

Further, we use the auxiliary variables for stating the route cost in the objective function. We delete the term $\sum_{m \in \mathcal{M}} u_m c^{\mathrm{r}}_m$ and modify the trip cost for the $T(m)$ instead. For an equivalent formulation, we use modified trip costs $\hat{c}^{\mathrm{t}}$ that are defined as

$$\hat{c}^{\mathrm{t}}_t = \begin{cases} c^{\mathrm{t}}_t + c^{\mathrm{r}}_{M(t)} & \text{if } t = (M \circ T)(t) \\ c^{\mathrm{t}}_t & \text{otherwise} \end{cases} \qquad \text{for } t \in \mathcal{T}.$$

We use the equivalent formulation with the modified cover constraints and the trip costs $\hat{c}^{\mathrm{t}}$ for implementing the optimal approach.

Instance Creation and General Procedure

The procedure for creating the test instances and providing them to the heuristical and the optimal solver is implemented in Python 2.7. The instances are created based on a test instance with realistic trips, provided by [Kai16] and [Kno16].

The following main files are called by the user in order to create a test instance and to solve it. The file `data.py` provides the functionality to create a test instance for a given number of customers, vehicles and a maximal number of alternative routes per customer. It calls the OpenTripPlanner (OTP) via a HTTP-request in order to create the multimodal routes and the OpenSourceRoutingMachine (OSRM) in order to compute a distance and time matrix. The file `build.py` is called to build the task graph. For each pair of trips, the time feasibility is examined. If applicable, a refuel point is picked for this arc and the arc fuel and arc cost are computed. The procedure to run the heuristics is contained in the file `heuristic.py`. As an argument, it requires the number of vertices per partial instance and the kind of heuristic to execute. After splitting the instance, it iteratively creates a partial task graph, invokes the solver for the partial instance, reads the partial solution and creates the endpoints. Finally, it composes an overall solution. For the iterative approach, it iteratively determines the customers to review and invokes the solver. The file `optimal_approach.py` is called to compute an optimal solution. It reads a feasible initial solution, computes the Pareto-optimal refuel points and calls the process for solving the instance to optimality.

The following auxiliary files contain the methods that are called by the main procedures. The file `otp.py` provides the methods to communicate with OTP. It opens the HTTP-connection and handles the requests. The file `osrm.py` opens the HTTP-connection to OSRM and creates the distance and time matrix for an instance. The file `instance.py` contains all information of a problem instance, i. e. the sets $\mathcal{C}$, $\mathcal{M}$, $\mathcal{T}$, $\mathcal{V}$ and $\mathcal{R}$. It further contains the distance and time matrix and the Pareto-optimal refuel points for each pair of trips. The file `solution.py` comprises a problem solution. Besides the information for vehicle duties and route choices, it contains methods to test the feasibility of the solution and to evaluate it. The methods to determine the estimated costs and the reviewed customers for the iterative approach, are contained in `iterative_approach.py`. The file `entities.py` contains all entities, i. e. trips, vehicles and refuel points. The file `taskgraph.py` provides methods to create the task graph and all kinds of partial task graphs. If further saves and loads the graphs to a file. The file `storage.py` contains methods to save instances, solutions and partial solutions to file and loads them.

Models and Heuristics

The model formulations that are used in the heuristics are implemented in the modeling language Mosel which is used by FICO Xpress Optimization Suite. The mixed-integer linear programs are realized just as defined in Chapter 4. The files `CMILP_i.mos` and `CMILP_1.mos` contain the (CMILP$_i$), the files `TMILP_i.mos` and `TMILP_1.mos` contain the (TMILP$_i$) for $i \geq 2$ and $i = 1$, respectively. For the iterative heuristic, the file `HSP.mos` contains the (HSP$_B$) and `MMILP_E.mos` contains both (FMILP$_S^{\max}$) and

(FMILP$_S^{\text{min}}$). The models read the structure of the task graph from a `.txt` file which has been created in `taskgraph.py` and write the computed solution to another `.txt` file that is loaded by `taskgraph.py` again.
In order to restrict the running times of the heuristics, for each of these problems a maximum time of is set. This means, if after this time no optimal solution for this partial instance is found, the algorithm terminates and returns the current best feasible solution. If no feasible solution is found up to then, the solver continues until he finds a feasible solution.

Optimal Approach

For the realization of the optimal approach, we use the framework DIP (cf. [RGW09]). This framework provides the core function for a Dantzig-Wolfe decomposition and is implemented in C++. In order to solve the problem optimally, we customize the framework by overriding the respective methods. We create the class `SchedulingDecompApp` which is derived from the class `DecompApp`. This class is represented in the following files: `SchedulingDecompApp.h` and `SchedulingDecompApp.cpp` contain the methods for creating the task graph, defining the model, generating the initial values and the main method for solving the subproblem. The file `SchedulingDecompSolveRelaxed.h` contains the heuristic for solving the subproblem and `SchedulingDecompSolveRelaxedBoost.h` solves the subproblem optimally. We further create the class `SchedulingAlgoPC` which is derived from the class `DecompAlgoPC` and contains the Branch-and-Bound procedure.
The file `module.cpp` provides the interface which is available in Python. To the method `Solve`, a problem instance and an initial solution is submitted. This is then converted into an internal representation in the files `instance.h` and `instance.cpp`. With this, the customized algorithm is executed and solves the problem to optimality. After termination, the solution is returned and saved to file.

7.2 Computational Results

We have implemented all heuristical approaches that we have discussed in Chapter 4 an the optimal approach as examined in Chapter 5. We evaluate these algorithms on different test instances. We vary different parameters in the instances as well as in the algorithms in order to appraise the solution methods in more detail. We expect the heuristics to cope with significantly larger input sizes than the optimal approach. Thus, we examine the heuristics as both initial solution for the optimal approach and solution method itself. In the former case, we choose the instance sizes such that an optimal solution can be expected in reasonable time. In the latter case, we aspire realistic instance sizes in order to estimate the behavior in a realistic environment.

According to the results of the underlying theses, we expect the optimal approach to cope with about 140 trips (cf. [Kai16, p. 139]) and the heuristics to cope with about 2000 trips (cf. [Kno16, p. 138]).

Test Instances

For testing the heuristical solution methods, we consider instances with 100 vehicles and 500 customers. We examine three cases, the number of alternative routes per customer is restricted to 2 in the first case, restricted to 3 in the seconds case and unrestricted in the third case. For each of these cases, we create 4 test instances independent from each other according to Chapter 6. Thus, in average there are approximately 1800, 2300 and 3200 trips in each instance, respectively. Additionally, there are 313 refuel points in each instance. All these instances cover a time range of 24 hours and the trips are distributed over time similar to Figure 6.1. Further, we aim to test the heuristics on instances with varied customer extension. Therefore, we create a copy of each test instance, where we shift each route up to 6 hours randomly in time.
For the optimal approach, we use a test instance with

$$|\mathcal{V}| = 10 \qquad |\mathcal{C}| = 50 \qquad |\mathcal{M}| = 150 \qquad |\mathcal{T}| = 230 \qquad |\mathcal{R}| = 313$$

that covers a time range of 24 hours. Out of this test instance, we create several subinstances, this means their customer sets are subsets of the first customer set. For each instance, we create an additional instance in which 2 alternatives are selected for each customer. For computing an initial solution, we apply the time-dependent heuristic with 30 vertices per partial instance.

Heuristics

As stated before, we have a total of 24 test instances. For each of these instances, we have applied the Successive Heuristic with a customer-dependent splitting and a time-dependent splitting. We have tested them with an aspired number of vertices of 100, 200 and 400 for each partial instance. The computations took place on the Compute-Server. All computations have been performed on a Linux-based operation system with 2.2GHz CPU and 128GB main memory.
For each execution of the heuristic, we measure the computation time. This time comprises creating the partial task graphs, solving the partial instances, determining the end points and connecting the partial solutions. It does not contain retrieving the distance and time matrix and building the task graph, from which the partial task graphs are created. We have not examined the determination of a lower bound for

the optimal objective value for this problem. Therefore we cannot come up with an optimality gap for these heuristics.

Most of the heuristic applications result in feasible duties. In some cases, a partial instance is infeasible. Then, the complete heuristic cannot be completed. In order to avoid these infeasibilities, we have ensured that the number of available vehicles is sufficient for the number of customers. Nevertheless, the first partial instance I_1 is infeasible in several executions of the heuristic with the following reason: Let $t \in \hat{\mathcal{P}}_1$ be an end point with initial fuel f_t^0. Due to (4.13) holds $f_t^0 \leq f_t^{\max}$ with $f_t^{\max} = 1 - \min_{r \in \mathcal{R}} \left(f_{r,t}^{\mathrm{d}} \right)$. Because of the simplification as described in Section 7.1, there is only one refuel point on each arc. We call this refuel point $R(s,t)$ for $(s,t) \in \overline{A}$. If the initial fuel of the end point t is too large, i. e. $f_t^0 > 1 - f_{R(v,t),t}^{\mathrm{d}}$ for all $v \in \mathcal{V}$, the end point cannot be connected feasibly to any vehicle and thus I_1 is infeasible. In order to avoid this malfunction, we could add a suitable refuel point $r \in \arg\min_{r \in \mathcal{R}} \left(f_{r,t}^{\mathrm{d}} \right)$ to each arc $(v,t) \in \overline{A}$ with $v \in \mathcal{V}$ and $t \in \mathcal{T}$. However, implementing this improvement requires several modifications in the structure of the heuristics. As the heuristic provides feasible solutions for each test instance anyway, this implementation is left open.

We have evaluated 24 test instances, where 12 of them emerge from shifting routes of the other instances. For each kind of instance, i. e. restricted to 2 or 3 alternatives or unrestricted, we have chosen 2 test instances each, whose total costs are in average, in order to provide their results in detail. Table 7.1 provides the results for these selected instances and shows in which components the total cost is split up. For each instance, the heuristic has been evaluated 6 times, i. e. 3 different splitting sizes with customer- and time-dependent splitting each. We provide the respective best result of these evaluations w. r. t. the total cost. The columns 1-4 state the instances sizes, the other columns show the total cost, the number of vehicles used and the respective trip, deadhead and route cost.

$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathcal{T}\rvert$	Total Cost	Duties	Trip	Deadhead	Route
100	500	996	1793	13,149,336	23	3,063,922	1,626,368	7,309,047
100	500	997	1828	13,641,367	26	3,398,177	1,518,512	7,424,678
100	500	1486	2327	12,171,599	21	2,840,081	1,033,910	7,247,609
100	500	1491	2338	12,392,594	20	2,876,440	1,332,836	7,183,318
100	500	2335	3190	11,327,191	17	2,103,052	1,207,107	7,167,032
100	500	2354	3228	11,364,031	18	2,114,112	1,034,166	7,315,753

Table 7.1: Distribution of the schedule cost for several instances

In order to examine the influence of the customer extension L_{C} to the heuristic in a realistic environment, we have created test instances with a large customer extension.

For more comparability, these instances emerge from the original instances by just shifting the routes in time. In the original instances, the average customer extension, i. e. $\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\left(z_c^{\text{end}} - z_c^{\text{start}}\right)$ is about 0:30 hours in each instance. By shifting the trips, the average customer extension grows to about 2:00 hours if each customer has at most 2 alternatives, 3:00 hours with 3 alternatives and 4:00 hours in the unrestricted case. We have evaluated all instances and the respective shifted instances with the same kind of heuristic. In summary, the total costs of the solutions are 3.1% higher in the shifted instance. If we consider only solutions which have been created with a customer-dependent splitting, the total costs are 2.6% higher, with a time-dependent splitting they are 3.5% higher. These results are based on 47 pairs of feasible heuristic solutions. We can see that both heuristics can cope with larger customer extensions. In this case, the results are slightly worse, where the customer-dependent heuristic handles the bad conditions better than the time-dependent heuristic.
Table 7.2 shows the results of one selected instance of each kind and its respective shifted instance. Columns 1-2 state the instance sizes, columns 3-4 the average and the maximum customer extension in hours. Columns 5-8 state the total cost and the number of duties for a customer-dependent and a time-dependent splitting. In each evaluation, the splitting size is 200 vertices per partial instance.

		Customer Extension		Customer-Splitting		Time-Splitting	
$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{T}\rvert$	avg [h]	max [h]	Cost	Duties	Cost	Duties
500	1793	0:28	2:38	13,369,843	24	13,149,336	23
500	1793	2:08	6:00	13,579,114	27	13,684,189	29
500	2327	0:32	1:55	12,445,088	19	12,385,898	21
500	2327	3:10	6:42	12,963,918	23	12,937,916	26
500	3190	0:37	2:22	11,448,625	17	11,447,670	16
500	3190	3:58	8:25	11,824,691	18	11,572,252	20

Table 7.2: Solution behavior with modified customer extension

Finally, we compare the heuristics w. r. t. their type of splitting. We evaluate all instances with a customer-dependent splitting and a time-dependent splitting and with 100, 200 and 400 vertices per partial instance. For all kinds of heuristics, we compare the total cost of the solution and the computation time of the heuristic. In summary, the total costs of the solutions that have been computed with the customer-dependent heuristic, are in average 0.9% higher than with the time-dependent splitting. Further the computation time is 45.0% higher. Comparing the splitting sizes, 200 vertices per partial instance provide the best results. The total costs for solutions that have been computed with 100 vertices are in average 1.9% higher, with 400 vertices 0.8% higher. The computation time for a splitting size of 200 is 3.18 times the computation time for splitting size 100, the computation time for splitting size 400 is 1.89 times the computation time for 200 vertices. These results are based on 29 pairs of feasible solutions for comparing the kind of heuristic and 18 triples of feasible solutions for comparing the splitting sizes. In summary, the time-dependent heuristic gives slightly better results than the customer-dependent heuristic and the computation time is significantly shorter. If we split the instance in smaller partial instances, the heuristic is much faster, but provides slightly worse results. If the size of the partial instances is too high, i. e. 400 vertices each, then the partial instances cannot be solved to optimality in the given time window of 10 minutes. Then the partial solutions are not optimal and therefore the heuristic solution is worse. We expect to receive a better heuristic solution for splitting size 400, if we increase the maximum computation time. In Table 7.3, we compare the different splittings on 3 exemplary instances, one for each kind of test instances. The instances have the following values:

I_1 :	$\lvert\mathcal{V}\rvert = 100$	$\lvert\mathcal{C}\rvert = 500$	$\lvert\mathcal{M}\rvert = 997$	$\lvert\mathcal{T}\rvert = 1828$	2 alternatives
I_2 :	$\lvert\mathcal{V}\rvert = 100$	$\lvert\mathcal{C}\rvert = 500$	$\lvert\mathcal{M}\rvert = 1486$	$\lvert\mathcal{T}\rvert = 2327$	3 alternatives
I_3 :	$\lvert\mathcal{V}\rvert = 100$	$\lvert\mathcal{C}\rvert = 500$	$\lvert\mathcal{M}\rvert = 2335$	$\lvert\mathcal{T}\rvert = 3190$	unrestricted

The columns 1-2 state the splitting type and the approximate number of vertices per partial instance. Columns 3-5 state the total cost of the solution in magnitude $\cdot 10^6$ for each test instance, columns 6-8 state the number of duties and columns 9-11 show the computation time in hours. In the customer-dependent heuristic with splitting size 100, applied on the test instance I_3, the first partial instance has been infeasible. Therefore, no heuristic solution exists.

Splitting		Total Cost [$\cdot 10^6$]			Duties			Comp. Time [h]		
		I_1	I_2	I_3	I_1	I_2	I_3	I_1	I_2	I_3
customer	100	13.84	12.83	-	26	21	-	0:14	0:16	-
time	100	13.74	12.47	11.59	23	19	17	0:10	0:12	0:29
customer	200	13.80	12.44	11.44	26	19	17	0:49	1:22	1:29
time	200	13.64	12.25	11.44	26	20	16	0:40	0:40	0:40
customer	400	14.28	12.38	11.59	40	21	18	0:57	1:45	2:54
time	400	14.56	12.17	11.32	42	21	17	0:53	1:27	2:20

Table 7.3: Comparison of different splittings

Iterative Heuristic

In the iterative heuristic, the route choices are reviewed. This approach improves an existing heuristic solution. Based on the existing solution, we identify customers with potentially bad route choices and review them iteratively. This approach requires several alternative routes for the customers, thus we examine it only on the test instances with 3 alternatives and an unrestricted number of alternatives. For each, we select 2 instances for testing, thus we test 4 instances in total. For comparability, we apply the iterative heuristic always on the solution, that has been computed with the time-dependent heuristic with 200 vertices per partial instance. We fix the parameters for the customer choice according to Algorithm 2 as follows:

$$r_{\min} \in \{1.5, 2, 3\} \qquad c_{\max} = 10 \qquad t_{\max} = 7200 \qquad n_{\max} = 10$$

This means, we vary only the minimal ratio for customers that are reviewed. The algorithm performs at most 10 iterations with at most 10 customers each. Thus, at most 100 customers may be reviewed. The time range for the subproblem is 2 hours, hence we can almost cover the complete instance of 24 hours in 10 iterations, if this is necessary.
Surprisingly, we do not recognize a different behavior of the iterative heuristic on the instances with restricted alternatives and the unrestricted instances. Based on the evaluations of 4 test instances, we receive the following results:

- $r_{\min} = 1.5$: In average, there are 77 customers whose ratio lies above the threshold. Out of these 77 reviewed customers, for 11 customers the route is changed. The objective value improves by 1.3% and the computation time is about 1:00 hour for the iterative heuristic without computing the initial solution.
- $r_{\min} = 2$: In average, 23 customers are reviewed and the route is changed for 4 of them. The total cost is decreased by 1.0% and the computation time is about 0:30 hours.
- $r_{\min} = 3$: In average, from 5 reviewed customers the route is changed for 1 customer. The total cost improves by 0.2% and the computation time is 0:15 hours in average. In contrast to the other evaluations, the restriction of the iterations is not exceeded here.

In the majority of iterations, there is no customer whose route is changed. Nevertheless, the total cost decreases in most of the iterations. This means that the Iterative Heuristic improves not only the route choices, but also the assignment of trips to vehicles and the visitation of refuel points.
Finally, we provide the evaluation results for one exemplary instance with $|\mathcal{V}| = 499$ and $|\mathcal{V}| = 2338$. The total cost of the initial solution amounts 12,392,594 with 20 duties. Table 7.4 states the results for 3 evaluations with different values for $r_{\min}$. The columns provide the total cost of the iterative solution, the number of vehicles

used, the number of reviewed customers, the number of customers with changed route decisions and the total computation time for the evaluation.

r_{min}	Total Cost	Duties	Reviewed Customers	Adjusted Customers	Comp. Time [h]
1.5	12,180,474	22	85	13	1:01
2	12,277,797	20	25	5	0:30
3	12,382,980	20	4	0	0:07

Table 7.4: Test results for the iterative heuristic

It is remarkable that the number of duties increases in the first evaluation. Instead, the deadhead cost decreases from 1,332,836 to 1,119,303.

Optimal Approach

We examine the optimal approach with an initial solution that has been computed with a time-dependent heuristic. Instead of just one possible refuel point in each arc, we consider a set of Pareto-optimal refuel points. This heavily increases the number of feasible duties and therefore the computation time. A great issue in the optimal approach is solving the subproblem, i. e. finding feasible vehicle duties with negative reduced cost. For this, we create a SPPRC with the resources reduced cost, fuel, duty length and routes. The number of labels in Algorithm 3 grows exponentially with the number of resources. As there is one resource for each route $m \in \mathcal{M}$, we expect a considerably increased computation time by the route resources. Thus, we examine whether including the route resources yields a better behavior of the algorithm. All computations have been performed on a notebook with Intel Core i5-6200U 2.3GHz CPU and 8GB main memory that runs on an Microsoft Windows 10 operation system. First, we apply the optimal approach without including the route resources. Table 7.5 provides the computational results for the sequence of test instances. The test instances in the first part have 2 alternatives per customer, in the second part 3 alternatives each. The maximum computation time is set to 30 minutes. After reaching the time limit, the algorithm returns the current best solution. The columns show the number of customers and trips in the test instance, the total cost of the initial solution and the optimal solution and the number of duties in the optimal solution. Further the gap between the initial solution and the optimal solution, the number of nodes in the Branch-and-Bound process and the computation time in minutes is shown.
For the respective last instances with $|\mathcal{T}| = 92$ and $|\mathcal{T}| = 69$, the algorithm does not have an optimal solution when reaching the time limit. There, the current best solution and the number of nodes that have been created so far is shown. It is noteworthy that the computation time of the largest solvable instances is still far away from the time

$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{T}\rvert$	Initial Cost	Optimal Cost	Duties	Gap	Nodes	Comp. Time [min]
10	32	284,281	284,202	1	0.03%	1	0:02
15	47	417,268	417,034	2	0.06%	1	0:04
20	61	602,907	602,889	2	0.00%	1	0:33
25	79	776,536	773,129	2	0.44%	3	4:33
30	92	826,880	826,358	3	0.06%	5	30:00
10	47	279,419	278,795	1	0.22%	1	0:11
15	69	404,437	403,174	2	0.31%	3	30:00

Table 7.5: Test results for the optimal approach

limit. While the instance with $\lvert\mathcal{T}\rvert = 79$ is solved to optimality in less than 5 minutes, the instance with $\lvert\mathcal{T}\rvert = 92$ cannot be solved in 120 minutes. Further, in each evaluation the number of duties is equal and the route choice is not changed for any customer in comparison to the initial solution. Only the deadhead cost is decreased during the algorithm.
As stated before, including the route resources results in a significantly larger number of labels and thus solving the subproblem requires more computation time. The benefit is that the column generation process does not add duties that solely violate the cover constraints. For the decision whether the route resources are used, this benefit has to be balanced to the enlarged computation time. We have evaluated the test instances with using the route resources. With this, no solution was computed in the given time limit. What is more, in the larger instances the algorithm stopped earlier as the available main memory was not sufficient.
As the optimal approach is only an enhancement of the already existing approach, we do not evaluate this approach in more detail. For a more extensive examination, particularly with test instances on smaller time windows and an analysis of the column generation and the Branch-and-Bound process, we refer to [Kai16, Sec. 10.2].

7.3 Interpretation of the Results

Finally, we link the computational results as provided in Section 7.2 with the application. Based on the information of Car2Go, on current day there are 155 electrically-powered vehicles in Stuttgart. They can be recharged at 313 stations. (cf. [Kai16, p. 144]) Of course, these vehicles do not drive autonomously. According to Figure 6.1, there are approximately 700 rentals per day on average. Each of these rentals stands for one customer in our problem setting. For each customer, we compute alternative routes whose start and finish positions and times are similar to the original rental. An

alternative may comprise more than one car trip, i. e. the customers uses a vehicle before and after public transport. We have created the sets $\mathcal{C}$, $\mathcal{M}$ and $\mathcal{T}$ based on these considerations.
In order to simulate a realistic problem sizes, we create a test instance with 150 vehicles, 698 customer and an unrestricted number of alternatives per customer. This instance consists of 3201 multimodal routes with 4355 trips. We solve this instance with the time-dependent heuristic and about 400 vertices per partial instance. In order to receive good results, we increase the maximal computation time per partial instance to 30 minutes. We receive a solution with 22 duties and a total cost of 15,498,352.
In Table 7.6 we see the distribution of the total costs and an evaluation of the times. In the first part, the trip, deadhead and route cost is presented with the magnitude $\cdot 10^3$. In the second part is listed how much time the vehicles spend for which activity. The trip time and deadhead time is the time that a vehicle is actually driving. The waiting time states the time that a vehicle waits between two trips without driving and at a refuel point when it is already fully charged. The time before the first trip and after the last trip are not included in the waiting time. The columns state the total cost and time, the average per duty and the average per customer.

	Total	per Duty	per Customer
Trip Cost	3,019	137.2	4.3
Deadhead Cost	1,125	51.1	1.6
Route Cost	10,254	466.1	14.7
Trip Time	70:44	5:56	0:11
Deadhead Time	25:20	1:09	0:02
Waiting Time	220:24	9:56	0:18
Refueling Time	91:01	4:08	0:07

Table 7.6: Cost distribution for a realistic instance

The heuristic solution contains 22 duties, i. e. we need 22 autonomous vehicles in order to satisfy 700 customer. This is one-seventh of the available fleet. We see that the vehicles cover 3000km carrying customers and 1100km for driving to the customers and refuel points. Each vehicle covers 188km in 24 hours. According to the considerations in Section 6.3, the average route costs are 7.30 euros per customer. This stands for the user inconvenience, i. e. a real cost for using public transport and an imaginary cost for spending time. A penalty cost of 50,000 occurs for each used vehicle, this means the cost for using an additional vehicle corresponds to a detour of 50km.
Considering the time, each vehicle spends in average 6 hours for carrying customers and 1 hour for deadhead trips. Further it is recharged 4 hours. The waiting time amounts 10 hours. We can see that the vehicle are not working to capacity. However, during the

peak there are more than 60 trips per hour. Thus, the majority of vehicles is needed to cover the trips at the peak. During the periods with less demand, the vehicles are idle. As we do not require the vehicles to be recharged after usage, the recharging time is only 4 hours. The range of a vehicle is 135km, with 4 hours recharging it increases to about 205km. This is only a little more than the actually covered distance of 188km. We can see that instances, where more alternatives are available, result in significantly better solutions. This behavior is caused by two reasons: On the one hand, each solution in an instance with fewer alternatives is also a feasible solution in an instance with more alternatives, but not vice versa. This obviously leads to a better solution for the instance with more alternatives. On the other hand, while creating the instances we prefer routes with two trips to routes with only one trip. We do this in order to include all facets of the problem setting in a relatively small instance. However, routes with two trips are not very profitable in general due to the restricted ability of the OTP. The later included alternatives are more profitable and thus the solution is better for instances with more alternatives.

Comparison to the Underlying Theses

Finally, we compare our computational results to the respective results of the underlying theses. [Kno16, pp. 135-136] have evaluated the Successive Heuristic for the simplified problem setting without cover constraints. They used realistic instances as shown in Table 7.7. It states the number of vehicles and trips, the number of partial instances, the optimality gap and the computation time in hours. The optimality gap has been computed with relaxing the fuel constraints.

$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$	Partial Instances	Gap	Comp. Time [h]
387	1836	7	7.30%	0:54
423	2031	6	2.63%	1:16
450	2400	7	9.61%	-

Table 7.7: Summarized results of [Kno16, Tables 10.3, 10.4, 10.5]

The time limit for the partial instances was 900 seconds. For the last instance, no computation time was given. In summary, the computation times as stated in Table 7.3 and Table 7.7 are similar for the same number of trips and the same size of the partial instances. In contrast to the problem setting without cover constraints, we cannot determine a lower bound with relaxing the fuel constraints. According to Theorem 3, our problem is still $\mathcal{NP}$-hard then.

[Kai16, pp. 137-140] have evaluated the Optimal Approach for the simplified problem setting with single-leg cover constraints. In Table 7.8, we compare the respective largest instances that could be solved to optimality in the time limit of 30 minutes. The upper section shows the previous results for single-leg cover constraints and the lower section shows the results for the multi-leg cover constraints. The columns show the number of customers, the number of trips, the number of trips per customer, the number of nodes of the Branch-and-Bound tree and the computation time in minutes. The computation conditions of both evaluations are similar.

$\|\mathcal{C}\|$	$\|\mathcal{T}\|$	$\|\mathcal{T}\|/\|\mathcal{C}\|$	Nodes	Comp. Time [min]
50	114	2.3	1	9:02
60	138	2.3	1	22:02
25	79	3.2	3	4:33
10	47	4.7	1	0:11

Table 7.8: Summarized results of [Kai16, Table 10.4] and Table 7.5

We see that the previous algorithm is able to cope with a far larger number of trips. Possibly, this is caused by an increased number of alternative trips per customer. The previous algorithm can solve instances with more trips, while the number of trips per customer is significantly lower than in our setting. This assumption is confirmed by the face that our algorithm is able to solve an instance with 79 trips and 25 customers, but not a similar instance wit 69 trips and 15 customers.

Chapter 8

Conclusion

In this final chapter, we give a summary of the problem setting and the developed solution methods for computing a heuristical and an optimal solution. We mention the crucial test results and give an overview of possible extensions for future work.

We regard the optimal integration of autonomous cars in car sharing. We assume a set of customers with certain travel requests which are known to us in advance. The customers are satisfied by multimodal transport, i. e. they use alternative routes that consist of car and public transport trips. The autonomous vehicles have a certain fuel range and the possibility to refuel. We aim a schedule of the vehicles such that for each customer one alternative is fulfilled and the fuel range is considered. This problem is classified as a Vehicle Scheduling Problem with resource constraints and cover constraints and is $\mathcal{NP}$-hard. The work is based on two previous master theses. [Kno16] and [Kai16] provide a heuristic and an optimal approach for a simplified problem. Our task was to adapt the heuristic and the optimal approach to the current problem setting and link both solution methods.

First of all, we develop an arc flow formulation for modeling the problem. It is based on a task graph with nodes for vehicles and trips and edges for feasible connections. Further the possibility to refuel is considered there. We create a mixed-integer linear program that defines a flow on the task graph satisfying the fuel constraints and the cover constraints. The goal is to minimize the total cost, consisting of costs for vehicles, trips, deadhead trips and routes.

In order to create a heuristic, we split the whole instance at some time points. Then we solve the emerging partial instances successively, beginning with last one. In each partial instance, we connect the duties to the duties of the previously solved partial instance. This provides a feasible schedule that fulfills the fuel constraints. For satisfying the cover constraints, we examine two approaches: In the customer-dependent splitting we enforce all trips of the same customer to be in the same partial instance and apply the cover constraints separately there. Unfortunately, this splitting cuts off feasible solutions. In the time-dependent splitting we may have trips of one customer in different partial instances. We select the routes by using a cost estimation instead of full information. Further, we use the Iterative Heuristic in order to improve a heuristic solution. Customers with potentially bad routes choices are identified and their route choices are reviewed iteratively. Evaluating these heuristics on several test instances, we learn that the heuristics with both splitting types lead to very good results for large instances sizes. The iterative approach further improves an existing solution in reasonable time.

In order to receive an optimal solution for the considered problem, we develop a path flow formulation. Here, the set of feasible duties for each vehicle is regarded. We apply Dantzig-Wolfe-Decomposition, where only the cover constraints are left in the master problem and the creation of feasible vehicle duties is referred to the subproblems. The subproblems are equivalent to a Shortest Path Problem with Resource Constraints where we introduce the reduced cost, fuel, duty length and fulfilled routes as resources. We use a label-setting algorithm in order to find a vehicle duty with negative reduced cost. With inverting the subproblem and searching paths from the end to the start, we solve all subproblems in just one execution. Further, we use both a heuristic and an optimal algorithm for generating new columns. Having a solution of the relaxed master problem, we enforce integrality by a Branch-and-Bound procedure. We use a heuristic solution as a starting solution for the column generation process. Applying the algorithm on test instances, we see that only very small instances can be solved to optimality.

Open Topics

Finally, we give an overview of conceivable further topics that are related to this thesis. This includes possible model extensions as well as further solution methods.

In order to make the model more realistic, it should be applicable on several days in a row. In the current setting, a vehicle starts with some initial fuel state, but it may be completely discharged at the end of a duty. In a more realistic model, a vehicle has the possibility to drive to a refuel point at the end of a duty and its fuel level at the end of the instance amounts at least the initial fuel level. Moreover in the current setting, the vehicles may drive to each refuel point all the time. To be more realistic, each refuel point has some capacity as there is only a restricted number of power sockets. Thus, only a certain number of vehicles can be charged at some refuel point simultaneously. The creation of the multimodal routes with more than one car trip has potential for improvement. The approach that is used here is only a transitional solution in order to generate suitable test instances. The generation of multimodal routes that are optimal in some sense stays open for further investigation.

An additional heuristic for the problem without cover constraints is based on Lagrangean Relaxation (cf. [Kno16, Cap. 9]). In this approach, the constraints connecting the partial instances are relaxed. In the customer-dependent splitting, all cover constraints occur in the partial instances. Thus, relaxing the constraints between the partial instances is similar to the problem setting without cover constraints. Thus, ap-

plying the Lagrangean Relaxation in combination with the customer-dependent splitting might be worth trying.
In the optimal approach, there are further possibilities that potentially improve the computation time. We have examined the route resources and observed that their direct application is not applicable as this yields to many combinations. On the other hand, these resources are not necessary for all routes and may be dropped after all trips of the respective customer are treated. If a meaningful usage of the route resources improves the algorithm, stays open for further research. Additionally, we can think of other methods to link the heuristics and the optimal approach. As the optimal approach performs better on smaller time windows (cf. [Kai16, Sec. 10.2]), applying the optimal approach only at the peak and continue the solution with heuristical methods might be a promising approach.

Previous Formulation

For sake of completeness, we provide the mixed-integer linear program of the underlying theses [Kai16, p. 34] and [Kno16, p. 34]. The notation is equivalent to the notation in Chapter 2 and Chapter 3. The only difference is the following: The mapping $C : \mathcal{T} \to \mathcal{C}$ states the respective customer for each trip. The set of multimodal routes $\mathcal{M}$ does not exist.

$$
\begin{aligned}
\min \quad & \sum_{s \in \mathcal{V}} \sum_{t \in \mathrm{N}^+_G(s) \setminus \{d^{\mathrm{e}}\}} x_{s,t} c^{\mathrm{v}} \\
& + \sum_{t \in \mathcal{T}} \sum_{s \in \mathrm{N}^-_G(t)} \left[x_{s,t} \left(c^{\mathrm{d}}_{s,t} + c^{\mathrm{t}}_{t} \right) + \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} \left(c^{\mathrm{d}}_{s,r} + c^{\mathrm{d}}_{r,t} - c^{\mathrm{d}}_{s,t} \right) \right] \qquad \text{(MILP)} \\
\text{s.t.} \quad & \sum_{t \in \mathrm{N}^-_G(s)} x_{t,s} = \sum_{t \in \mathrm{N}^+_G(s)} x_{s,t} && \text{for all } s \in V \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\} \\
& \sum_{s \in \mathrm{N}^-_G(t)} x_{s,t} = 1 && \text{for all } t \in \mathcal{V} \\
& \sum_{t \in C^{-1}(c)} \sum_{s \in \mathrm{N}^-_G(t)} x_{s,t} = 1 && \text{for all } c \in \mathcal{C} \\
& \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} \leq x_{s,t} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) \\
& e_s \leq f^0_s && \text{for all } s \in \mathcal{V} \\
& 0 \leq e_s - \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{s,r} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) \\
& e_t \leq 1 - f^{\mathrm{t}}_t - \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} f^{\mathrm{d}}_{r,t} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) \\
& e_t \leq e_s - x_{s,t} \left(f^{\mathrm{d}}_{s,t} + f^{\mathrm{t}}_t \right) - \sum_{r \in \mathcal{R}_{s,t}} z_{s,r,t} \left(f^{\mathrm{d}}_{s,r} + f^{\mathrm{t}}_r + f^{\mathrm{d}}_{r,t} - f^{\mathrm{d}}_{s,t} \right) + (1 - x_{s,t}) \\
& && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t) \\
& x_{s,t} \in \{0,1\} && \text{for all } (s,t) \in A \\
& z_{s,r,t} \in \{0,1\} && \text{for all } t \in \mathcal{T}, s \in \mathrm{N}^-_G(t), r \in \mathcal{R}_{s,t} \\
& e_s \in [0,1] && \text{for all } s \in V \setminus \{d^{\mathrm{s}}, d^{\mathrm{e}}\}
\end{aligned}
$$

List of Figures

List of Tables

List of Algorithms

Bibliography

[Adl14] Jonathan D. Adler. *Routing and Scheduling of Electric and Alternative-Fuel Vehicles*. PhD thesis, Arizona State University, 2014.

[BCG87] A. A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17.3:271–281, 1987.

[BK09] Stefan Bunte and Natalie Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1.4:299–317, 2009.

[Car] Car2Go. Car2go.

[CL07] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153.1:29–46, 2007.

[DF54] G. B. Dantzig and D. R. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly*, 1.3:217–222, 1954.

[DP95] J. Daduna and J. Pinto Paixão. Vehicle scheduling for public mass transit - an overview. *Computer-Aided Transit Scheduling*, 430:76–90, 1995.

[FP95] R. Freling and J. Pinto Paixão. Vehicle scheduling with time constraint. *Computer-Aided Transit Scheduling*, 430:130–144, 1995.

[Hau15] Jan Hauser. Amerika schaltet auf autopilot. *Frankfurter Allgemeine Zeitung*, 2015.

[HZ80] G. Y. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10.4:293–309, 1980.

[ID05] S. Irnich and G. Desaulniers. *Column Generation*, chapter Shortest Path Problems with Resource Constraints, page 33–65. Boston, MA: Springer US, 2005.

[Kai16] Marcus Kaiser. Optimal integration of autonomous vehicles in car sharing: A decomposition approach in consideration of multimodal transport. Master's thesis, Technische Universität München, 2016.

[Kno16] Martin Knoll. Optimal integration of autonomous vehicles in car sharing: A decomposition approach and fastening heuristics. Master's thesis, Technische Universität München, 2016.

[LK81] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11.2:221–227, 1981.

[OSR] OSRM. Open source routing machine.

[OTP] OTP. Open trip planner.

[Raf83] S. Raff. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10.2:63–211, 1983.

[RGW09] T. Ralphs, M. Galati, and J. Wang. Decomposition for integer programming, 2009.

[VVS] VVS. Verkehrs- und tarifverbund stuttgart.

[WLR$^+$16] M. Wen, E. Linde, S. Ropke, P. Mirchandani, and A. Larsen. An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers and Operations Research*, 76:73–83, 2016.

List of Corrections