A Branch-and-Price Algorithm for Optimal Routing of Delivery Robots

Stefan Schaudt[1], Uwe Clausen[2], and Nicklas Klein[3]

[1]TU Dortmund University, Dortmund, 44227, Germany, stefan.schaudt@tu-dortmund.de
[2]TU Dortmund University, Dortmund, 44227, Germany, uwe.clausen@tu-dortmund.de
[3]University of Bern, Bern, 3012, Switzerland, nicklas.klein@unibe.ch

April 27, 2022

Abstract

Increasing parcel volumes, environmental challenges, and customer expectations have made innovative delivery concepts for last-mile logistics essential in recent years. Autonomous delivery vehicles such as delivery robots or drones are tested worldwide. In this work, we examine the use of electrical delivery robots to optimize a last-mile network. The network consists of a fleet of homogeneous single-unit capacity robots, depots equipped with recharging stations, and customers. Each customers is defined by a time window and a profit. The goal is to find a set of tours that maximizes the total collected profit. These tours have to respect the customers' time windows and battery constraints of the robots. We present a branch-and-price algorithm to solve this combinatorial optimization problem exactly. Within this algorithm, the problem of finding feasible tours arises. To decide on the feasibility of a tour, we present a polynomial dynamic program and prove its correctness. The computational studies on modified benchmark instances show that the algorithm can solve instances that have realistic time window lengths and up to 144 customers in a reasonable time.

Keywords: Branch-and-Price, Team Orienteering, Vehicle Routing Problem, Multiple Depots, Partial Recharging, Electric Vehicle Routing Problem, Exact Algorithm, Delivery Robots, Dynamic Program, Last-Mile

1 Introduction

The logistics industry has been facing significant challenges in recent years. These challenges include an increasing parcel volume and the customers' expectation to act environmentally friendly. In 2019 the total number of parcels in the 13 major markets surpassed the 100 billion mark for the first time, which is an increase of 17.7 % compared to the previous year [2]. To meet the customer expectations, the logistic industry strives to operate more sustainable and environmentally friendly. Therefore, logistics companies announced zero-emissions targets. DHL, for example, plans to reduce all logistics-related emissions to zero by 2050 [9]. To overcome these challenges, the logistics industry explores new innovative, sustainable concepts for parcel and good deliveries to overcome these challenges. One concept to perform last-mile logistics is based on small electric autonomous vehicles, such as drones or delivery robots [4]. Compared to traditional trucking, these concepts have several advantages; first, a reduction of pollution due to the electrical engines. Secondly, an increase of flexibility for logistics service providers due to

the autonomy, and lastly, a reduction of road traffic since both vehicle types do not operate on streets. However, the concepts differ in their application fields. Drones are incredibly well suited for low-density areas, as they can cover large distances in a short time and, due to the use of airspace, are not dependent on the road network. Using delivery robots is especially well suited for urban areas, where the robots can carry small goods over short distances using sidewalks or pedestrian zones. Both concepts allow for time window-based deliveries and a contactless handover of goods.

To better understand the problem discussed in this contribution, we describe the delivery process of parcels with robots in the following. We assume that a fleet of robots is located at micro-depots close to some customer locations. To increase service level and decrease the number of failed delivery attempts, each customer chooses a delivery time window in advance. Before the first delivery takes place, a delivery schedule is calculated, and the parcels are distributed from a hub to the calculated micro-depots by trucking. At the micro-depot, an available robot has to pick up the parcel and starts its journey to arrive at the customer's location within its chosen time window. Once the robot arrives, the customer is informed and opens the robot's compartment with a code or an application. After this service is completed, the robot has to travel back to a micro-depot to pick up another parcel. At the micro-depot, it can either recharge its battery or immediately start its next delivery. It is worth noting that the general method of this contribution is not limited to the delivery with robots but can be used for any transport concept with a single-unit capacity. Therefore, we use the word vehicle instead of delivery robot or drone in the following.

In contrast to fuel-powered vehicles, electric vehicles have a smaller range, and recharging is time-consuming. Especially for small electric vehicles, the battery capacity limits the deployability in delivery networks. Therefore, the battery capacity and recharging times should be considered when modeling the delivery process. Moreover, the assignment between delivery vehicles and micro-depots is assumed to be flexible, and a vehicle can pick up goods and conduct recharges at any micro-depots, making the network more flexible and robust.

The remainder of this paper is structured as follows: in Section 2, related literature is presented. Section 3 is dedicated to a detailed description of the considered problem. In Section 4, we present a dynamic program, which can solve the question of whether a single vehicle can visit a given sequence of customers such that time windows are satisfied and the state of charge remains within the bounds. The program also decides which micro-depot(s) to visit between two customers and how long to recharge at the micro-depot(s). The dynamic program is used within a branch-and-price algorithm, which is described in Section 5. In Section 6, computational results for the branch-and-price algorithm on benchmark instances are presented. The last section concludes and gives an outlook about potential future research.

2 Literature Review

In the following, we overview the literature related to our problem. This paper is the first work that considers the optimization of a delivery network with unit capacitated vehicles and partial recharges to the best of our knowledge. Therefore, our literature review mainly covers two fields of research: the delivery with autonomous vehicles, and variants of the vehicle routing problem.

A recent overview on innovative concepts for last-mile delivery is provided by Boysen et al. [4]. The authors discuss today's concepts, such as human-driven delivery vans or cargo bikes, and near and further future concepts, such as delivery robots or drones. The consideration of small autonomous delivery vehicles has created various problem formulations. In the following, we provide an overview of related literature concerning drones. A literature review on this topic is provided by Viloria et al. [26]. Many publications in this field investigate the combination

of a delivery vehicle with drones. In such a scenario, the vehicle functions as a mobile depot that not only picks up parcels but also offers a takeoff and landing place for drones ([1], [17]). This stands in contrast to our problem, where the classical truck is excluded from the problem formulation, and only stationary depots are considered. In addition, most publications on drones either do not consider the battery or only map it by a maximal flying distance, where recharges are assumed to be instantaneously by swapping the battery. There are only a few publications that consider the recharging process of the battery. For example, Mao et al. [15] investigate a location-routing problem for drones. The problem is motivated by reconnaissance missions, where drones have to visit targets within a given time window and spend reconnaissance time at the targets. The authors present a mixed-integer nonlinear programming formulation and an adaptive large neighborhood search algorithm.

Besides drones, the concept of delivering small goods with autonomous delivery robots has been developed in recent years. The number of publications considering delivery robots is relatively small. A case study based on a combined simulation and optimization approach is presented by Poeting et al. [19]. The authors simulate and optimize a parcel delivery network involving robots in a city in Germany. Their goal is to reduce the number of robots while maintaining a certain service level. In contrast to our problem formulation, the robots cannot change their micro-depot during the day, and recharging is not considered. Sonneberg et al. [22] consider a location-routing problem with the use of multi-compartment robots and customer time windows. Boysen et al. [3] examine the combination of parcel robots with a delivery van that functions as a mothership for the robots, similar to the drone approaches.

During the literature review, we found a lack of research on the consideration of charging processes for small autonomous delivery vehicles. In many publications, the charging process is modeled by a battery swap, and thus, the battery only limits the maximal tour length. However, since the recharging of batteries is time-consuming and battery swaps are not always possible, we include this process in our optimization model.

The Electric Vehicle Routing Problem (EVRP) is a related field where recharges are modeled. In the EVRP, a fleet of electric vehicles serves a set of customers. The fleet can recharge its batteries at designated recharging stations. An overview of the EVRP is given by Erdelić and Carić [10]. The EVRP with time windows was first investigated by Schneider et al. [21]. They derive a mixed-integer linear programming (MILP) formulation and present a combination of a Variable Neighborhood and Tabu Search heuristic to solve the problem. However, in their studies only full recharges are possible. This restriction is dropped by Keskin and Çatay [13], where partial recharges are considered. The authors present a MILP formulation and an Adaptive Large Neighborhood Search to solve the arising optimization problem. Desaulniers et al. [7] investigate different variants of the EVRPTW, including full and partial recharges. The optimization problem is solved with an exact branch-price-and-cut algorithm.

The problem we focus on is closely related to the EVRPTW with two main differences: firstly, the capacity of electric vehicles in terms of battery and cargo space is much higher than autonomous delivery vehicles. Hence, visiting a recharging station between two customers is optional in the EVRPTW. However, for small delivery vehicles, a depot visit is mandatory between two customers due to the single-unit capacity, which leads to a different modeling approach. Secondly, the EVRPTW requires that each customer is served within his time window and the goal is to minimize the distance traveled. The problem considered in this work is motivated by profit collection. Each customer has a profit assigned, collected when served within the respective time window, and not every customer has to be served. The variant of the Vehicle Routing Problem, considering this objective, is called the Team Orienteering Problem (TOP). Tae and Kim [24] consider a TOP with time windows and derive a branch-and-price algorithm to solve it. However, the authors do not consider electric vehicles or recharging processes. A

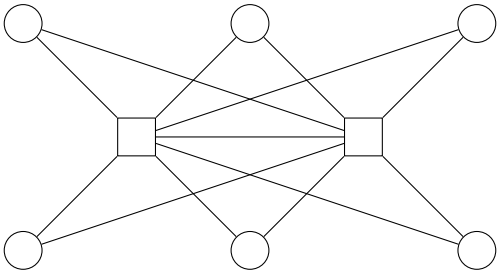


Figure 1: Example of a graph structure with six customers (circles) and two depots (squares).

capacitated TOP with time windows has been considered by Park et al. [18]. The authors propose a branch-and-price algorithm similar to the previous publication to solve the problem.

3 Problem Description

This section gives a formal description of the delivery problem with small autonomous robots, including partial recharges. Let $G = (V, E)$ denote a graph, where $V = C \cup D$ is the vertex set, with customer set $C = \{1, \dots, n\}$ and depot set $D = \{n + 1, \dots, n + d\}$. The edge set $E = \{\{i, k\} : i \in V, k \in D\}$ contains edges between depots and between depots and customers. Each customer $i \in C$ has a single-unit demand, a profit p_i, a time window $[l_i, u_i]$ within which the service should commence, and requires service of duration τ_i. Let $c_{i,k}$ denote the required time to traverse edge $\{i, k\} \in E$. The travel times are symmetric, so $c_{i,k} = c_{k,i}$ holds for each $\{i, k\} \in E$. The depot with the smallest travel time to customer $i \in C$ is denoted by $k(i)$ and we define $c_i = c_{i,k(i)}$. Furthermore, we are given m vehicles to operate. All vehicles have a single-unit capacity, so a depot visit is required between each pair of customers. Note that multiple depot visits are allowed between two customers. The assignment of customer orders to depots is not given beforehand and is solved within the optimization. All vehicles drive at a fixed speed and have a maximum battery capacity of B travel time units. The vehicle can recharge its battery at any depot $k \in D$. The state of charge (SoC) decreases linearly while driving and increases linearly while recharging. This simplification is needed, although in real-world applications, the charging rate decreases for the last 10% to 20% of the battery capacity (Marra et al. [16]). Moreover, using the capacity of a battery ultimately decreases its lifetime. Therefore, we assume that the actual physical capacity of the battery is not used completely, which justifies the linearity assumption. The recharging rate is given by α, whereas one time unit of charging results in α travel time units.

The goal of the optimization problem is to maximize the total profit collected such that

(i) each customer is visited at most once by any of the m vehicles,

(ii) each selected customer i is served within its time window $[l_i, u_i]$, and

(iii) the SoC of the vehicles should remain within the bounds $[0, B]$ at all times.

Note that it is not mandatory to visit all customers in this problem. It is assumed that the reduced sub-graph of depots is connected concerning the maximal travel time B, and each customer can be reached from at least one depot with a travel time not larger than $B/2$.

A compact overview of all notations can be found in Table 1. In addition, an abstract sample graph of this problem with six customers (circles) and two depots (squares) is displayed in Figure 1.

C	set of customers, $C = \{1, \ldots, n\}$
D	set of depots, $D = \{n+1, \ldots, n+d\}$
m	number of available vehicles
p_i	profit of customer $i \in C$
$[l_i, u_i]$	time window of customer $i \in C$
t_i	arrival time at customer or depot $i \in V$
T	time horizon
τ_i	service time at customer $i \in C$
$c_{i,k}$	travel time between customer or depot $i \in V$ and depot $k \in D$
c_i	travel time from customer $i \in C$ to its nearest depot (k(i))
B	battery capacity in travel time units
b_i	SoC when arriving at customer or depot $i \in V$
r_k	recharging time at depot $k \in D$
α	charging rate (travel time units per unit charged)
s	customer sequence

Table 1: General Notations

Before we present an algorithm for the stated problem, we describe a dynamic program to solve a variant of this problem. In this variant, a single vehicle and a fixed customer sequence are given. The task is to find a depot path between every two customers and to decide on the recharging time at the depots.

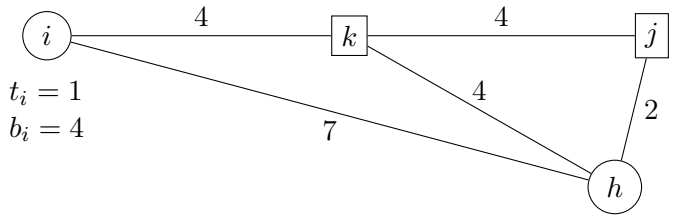
4 Dynamic Programming Algorithm

This section is devoted to the presentation of a dynamic program that is capable of solving the following decision problem: given a set of depots, a single vehicle, and a sequence of customers; is there a feasible path of depots between each pair of customers such that the battery remains within the bounds and customers' time windows are satisfied? Before describing the dynamic program in detail, we consider the related literature to this decision problem. A related problem is characterized by Tseng et al. [25]. The authors consider a variant of the TSP with a drone and recharging stations. Their goal is to minimize the total time for traveling and recharging. They enumerate over all feasible tours and select the shortest to solve this problem. A tour is defined as a given sequence of the customers and recharging stations. Moreover, the authors show that considering only the distance is sufficient to find the optimal tour. In their work, the authors show that recharging times can be considered implicitly through the feasibility of a tour.

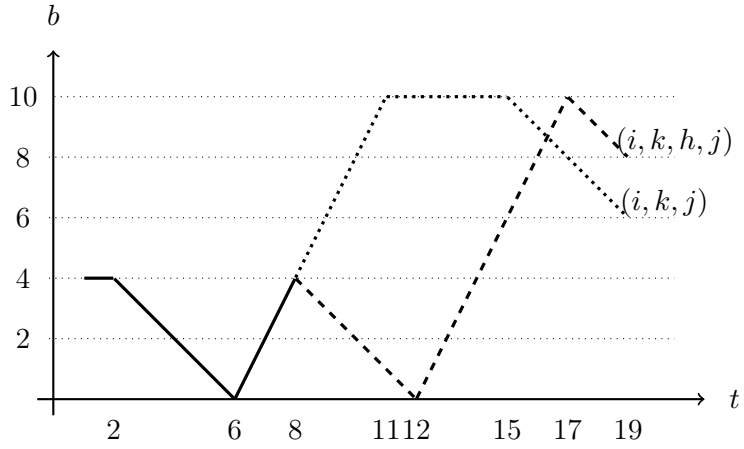
The same solution approach is used by Pourazarm et al. [20]. The authors consider the problem of finding a time-shortest path from a source to a sink node for electric vehicles through a network of charging stations. Similar to the previous problem, the objective is to minimize the sum of traveling and recharging times. Using the proportional dependency between traveling and recharging times reduces this problem to the shortest path problem, where only the traveled distance is minimized. However, such a reduction is not possible for our problem due to the time windows.

Sweda et al. [23] investigate optimal recharging policies for predetermined paths. The objective is to minimize costs composed of a stopping charge at a station, a recharging price per unit of energy, and overcharging costs if the SoC surpasses a certain level. They provide exact, efficient solution methods for different variants of the problem and heuristics.

To the best of our knowledge, the problem of finding depot paths combined with an optimal recharging policy for a sequence of customers with time windows has not been considered yet.



(a) Example graph with two customers and two depots.



(b) SoCs over time for the two alternative depot paths.

Figure 2: Example for different depot paths.

In the subsequent subsection, we present an example for the described problem (Example 1), define a feasible solution, and introduce the concept of *time-battery curves*. A time-battery curve maps arrival times to SoCs at a customer. In Subsection 4.2 the dynamic program is presented and its correctness is proven.

4.1 Preliminary Remarks

In the previously presented problem, a depot path between each customer pair needs to be selected and a decision on the recharging times at the depots is required. We can assume that a depot path does not contain a depot twice between two sequential customers. If a depot is visited twice, the second visit has a later arrival time, and if the time difference between the two visits is used for recharging, the second visit cannot arrive with a higher SoC. Therefore, all depots in between this circle can be simply discarded. In the calculation of a depot path, the question arises of which depot(s) to choose and whether all depots are equally "good". One might think that for a pair of customers, the depot that minimizes the total distance between them should always be chosen. Without a limit on the maximal SoC, this idea is optimal and minimizes the overall time spent. However, with a limit, the simple Example 1 shows that the distance minimizing depot is not always preferable.

Example 1. *Consider a given sequence of customers, where the vehicle is currently at customer i, and the next customer is j. The available depots are k and h. The required travel times, which are equal to battery consumption, are displayed next to the edges in Figure 2(a). In this example, the distance minimizing depot between customers i and j is depot k with a total travel time of eight units.*

The time window of customer j is $[19, 24]$. It is assumed that customer i is served at time one, and the service requires one unit. The vehicle's current SoC is four, and the battery capacity is 10. At customer i, the vehicle can only travel towards depot k, since depot h is not reachable with an SoC of four. Therefore, the vehicle arrives at depot k at time six with an empty battery. At the depot, the vehicle can recharge its battery and gains two driving units for each unit charged ($\alpha = 2$). Two options are available from depot k: the vehicle can travel directly to customer j or visit depot h first.

- *If a direct trip to customer j is chosen, the vehicle has to charge at least two units to reach j. Since the time window of the customer starts at time 19, the vehicle can charge its battery completely in five time units. However, to arrive at j within the time window, the vehicle has to remain idle at the depot for additional four units. It departs from depot k at time 15 and arrives at customer j at time 19 with an SoC of six. For this option the SoCs over time are displayed by the dotted line (i, k, j) in Figure 2(b).*
- *If we opt to visit depot h first, the vehicle must charge two time units. It leaves depot k at time eight and arrives at h at time 12 with an empty battery. The vehicle has to wait for additional five units to arrive on the lower time window bound. This time can be used to charge the battery completely. Afterward, it leaves the depot and reaches j at time 19, with an SoC of eight units. The SoCs over time are represented by the dashed line (i, k, h, j) in Figure 2(b).*

In summary, both options arrive at the earliest possible time, but the second option has a higher SoC, which is preferable. This example shows that the distance minimizing depot between two customers is not always optimal, and visiting multiple depots can lead to a preferable solution.

In general, the problem has two contrary goals; firstly, to minimize the total time, which gives a temporal leeway to meet upcoming time windows, on the one hand, and secondly, to maximize the SoC, which allows for longer trips without recharging, on the other hand. This conflict is resolved in Subsection 4.2.

In the following, we give a formal definition of a feasible solution, and define a function that maps arrival times and SoCs of the vehicle at a customer. A solution π for a given customer sequence s consists of a depot path for each customer pair with recharging times and a depot visit before the first and after the last customer of the sequence.

Definition 1. *A solution π is stated feasible if the SoC remains within $[0, B]$ and each customer $i \in s$ is served within its time window $[l_i, u_i]$.*

A key concept used in our algorithms and theorems is the so-called *time-battery curve*. The dynamic program uses it to map arrival times and SoC pairs at a customer $i \in C$.

Definition 2. *Given pairs $(\underline{t}_i^1, \underline{b}_i^1), (\underline{t}_i^2, \underline{b}_i^2), \ldots, (\underline{t}_i^q, \underline{b}_i^q)$ and pairs $(\bar{t}_i^1, \bar{b}_i^1), (\bar{t}_i^2, \bar{b}_i^2), \ldots, (\bar{t}_i^q, \bar{b}_i^q)$ with*

- *$\underline{t}_i^f \leq \bar{t}_i^f$ and $\underline{b}_i^f \leq \bar{b}_i^f$ for all $f = 1, \ldots, q$,*
- *$\bar{t}_i^f < \underline{t}_i^{f+1}$ and $\bar{b}_i^f = \underline{b}_i^{f+1}$ for all $f = 1, \ldots, q-1$,*
- *$l_i \leq \underline{t}_i^1$ and $\bar{t}_i^q \leq u_i$, and*
- *$c_i \leq \underline{b}_i^1$ and $\bar{b}_i^q \leq B - c_i$.*

A time-battery curve is a function $b : [l_i, u_i] \to [c_i, B - c_i]$ with

$$b(t) = \underline{b}_i^f + \alpha(t - \underline{t}_i^f), \text{ for all } t \in [\underline{t}_i^f, \bar{t}_i^f] \text{ and } f \in \{1, \ldots, q\}.$$

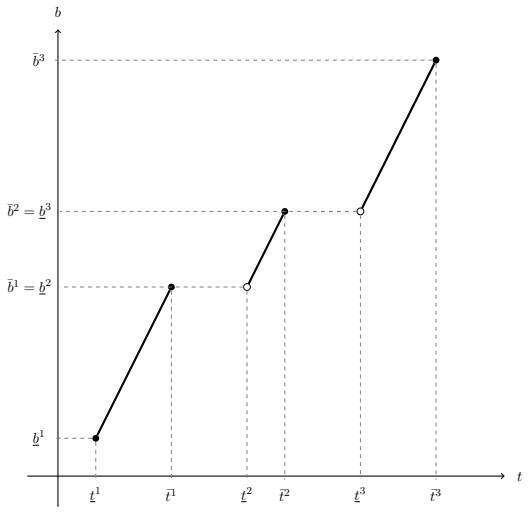


Figure 3: Example of a time-battery curve with three curve elements

The idea of this curve is to map combinations of arrival times and SoCs from different depots at a customer. This mapping is necessary since different depot visits lead to other time-battery options, as we have seen in Example 1. An example of a time-battery curve is displayed in Figure 3. This curve consists of three linear curve elements, which are defined by pairs $(\underline{t}_i^1, \underline{b}_i^1), (\underline{t}_i^2, \underline{b}_i^2), (\underline{t}_i^3, \underline{b}_i^3)$ and pairs $(\bar{t}_i^1, \bar{b}_i^1), (\bar{t}_i^2, \bar{b}_i^2), (\bar{t}_i^3, \bar{b}_i^3)$. To give a vivid interpretation of this curve, we assume that a customer is approached from the depots. We calculate two arrival time pairs and two SoC pairs for each of the three depots. The first pair represents the earliest possible arrival time and the corresponding SoC. The second pair is determined by the arrival time after the longest possible recharge at the depot. This recharging time is either given by a full recharge or is restricted by the upper time window bound of the customer. The time-battery curve aims to capture all these options. Moreover, it captures all arrival times and SoC combinations between the first and second pair for each curve element. Furthermore, the earliest SoC value of a curve element should be at least equal to the highest SoC value of the preceding element since arriving later at a customer with the same SoC is not favorable. The first pair is denoted by $(\underline{t}, \underline{b})$ and is called lower time-battery pair in the following. The second pair is denoted by $(\bar{t}, \bar{b})$ and is called upper time-battery pair in the following.

Observation 1. *It holds that the slope of the linear interpolation between two points on a time-battery curve is bounded by* α*.*

This observation states that a later arrival time at a customer can not lead to an SoC that is higher than an SoC from an earlier arrival if the time difference is used to recharge.

4.2 Dynamic Program for a Given Customer Sequence

This subsection is devoted to the dynamic program, which solves the previously stated decision problem in polynomial time. Besides the decision problem, the program can be easily extended, such that a feasible solution is returned if one exists. The necessary extensions are described in Appendix B. The dynamic program is later used to solve the pricing problem of our branch-and-price algorithm in Section 5. We assume that the vehicle starts its tour with a fully charged battery in the following.

Algorithm 1: Dynamic program to solve the decision problem.

Input: customer sequence $s = (i_1, \ldots, i_j)$, depot set D

1 $(\underline{t}^1_{i_1}, \underline{b}^1_{i_1}), (\bar{t}^1_{i_1}, \bar{b}^1_{i_1}) \leftarrow (\max\{l_{i_1}, c_{i_1}\}, B - c_{i_1})$
2 **if** $\underline{t}^1_{i_1} > u_{i_1}$ *or* $\underline{b}^1_{i_1} < \frac{B}{2}$ **then**
  Return: infeasible
3 **end**
4 **for** $k = 2$ *to* j **do**
5 call *Algorithm 2* extension from i_{k-1} to i_k
6 **if** *extension not possible* **then**
    Return: infeasible
7 **end**
8 **end**

Return: feasible

The dynamic program evaluates the customer sequence front to back, and at each customer, it captures all possible paths to arrive at the customer in a time-battery curve. It repeatably performs the following steps. From a customer, it calculates the earliest arrival time at each depot. Thereby, not only direct connections are considered but also the option of depot paths. Afterward, it calculates the lower and upper time-battery pairs from the depots at the next customer. Based on these pairs, it constructs a time-battery curve. A pseudocode of the dynamic program is given in Algorithm 1.

The input of Algorithm 1 is a customer sequence $s = (i_1, \ldots, i_j)$ and a set of depots D. For both algorithms, we assume that all other parameters such as the battery capacity B, distances, or the recharging rate α are defined globally. The algorithm's output is whether or not the customer sequence s is feasible. In the initialization, the earliest arrival time and SoC at the first customer i_1 is calculated, see line 1. Next, a subroutine that tries to extends from the current customer to the next customer of s is called, see lines 4 to 8.

The subroutine considers two sequential customers i and j and calculates the shortest depot paths between these customers. The input of Algorithm 2 are two customers $i, j \in s$, the depot set D, and a time-battery curve of customer i. A pseudocode of this algorithm is displayed in Algorithm 2. The following paragraph describes the pseudocode of Algorithm 2 in detail.

In the first part of the algorithm (lines 3 to 9), each depot that is directly reachable from customer i is evaluated, starting earliest with a sufficient SoC to reach the depot. The service time τ_i is added to the calculated arrival time. Since the time-battery curve at i may include horizontal gaps, the previously calculated direct path to each depot might not be the earliest possible arrival time when considering multiple depot visits. In the next part of the algorithm (lines 10 to 21), the earliest arrival time at each depot (including previously not evaluated depots) through an adapted shortest path search with respect to the SoC is calculated. Recharges are only conducted if the vehicle is not able to reach the next depot. Moreover, the recharging time is kept as small as possible.

The earliest and latest arrival time and the corresponding SoCs at customer j from each evaluated depot are calculated in the next step. If no depot is evaluated, the algorithm returns that no extension is possible (lines 23 and 32). Note all idle times at the depots are used to recharge the battery. Finally, the calculated values are sorted by the earliest time and in case of a tie by the SoC value (line 30). This order fits the structure of the time-battery curve that favors earlier arrival times and higher SoCs. Therefore, it is possible to consider every two values of a depot only once in the creation the time-battery curve at customer j. Furthermore, this order allows detecting if an extension from i to j is feasible.

Algorithm 2: Extension from customer i to customer j

Input : customers i, j, depot set D, time-battery curve $(\underline{t}_i^f, \underline{b}_i^f), (\bar{t}_i^f, \bar{b}_i^f), f = 1, \ldots, q$

$F \leftarrow \emptyset$
$t_k \leftarrow \infty, b_k \leftarrow 0, \forall\ k \in D$
# evaluate depots from customer i
for $k \in D$ *with* $c_{i,k} \leq \bar{b}_i^q$ **do**
  $(t, b) \leftarrow \min\{(\underline{t}_i^f, \underline{b}_i^f) : \bar{b}_i^f \geq c_{i,k}, f \in \{1, \ldots, q\}\}$
  $r \leftarrow \frac{1}{\alpha} \min\{0, c_{i,k} - b\}$
  $t_k \leftarrow t + r + \tau_i + c_{i,k}$
  $b_k \leftarrow b + \alpha r - c_{i,k}$
end
# depot to depot evaluation
while $F \neq D$ **do**
  $k \leftarrow \arg\min\{t_w : w \in D \setminus F\}$
  $F \leftarrow F \cup \{k\}$
  for $h \in D \setminus F$ *with* $c_{k,h} \leq B$ **do**
    $r \leftarrow \frac{1}{\alpha} \max\{0, c_{k,h} - b_k\}$
    if $t_k + r + c_{k,h} < t_h$ **then**
      $t_h \leftarrow t_k + r + c_{k,h}$
      $b_h \leftarrow b_k + \alpha r - c_{k,h}$
    end
  end
end
$D' \leftarrow \{k \in D : t_k \leq u_j, c_{j,k} + c_j \leq B\}$
if D' *is empty* **then**
  Return: extension not possible
end
calculate lower and upper time-battery pairs at customer j
$\underline{t}_k \leftarrow \max\{l_j, t_k + \frac{1}{\alpha} \max\{0, c_{j,k} + c_j - b_k\} + c_{j,k}\}$, for $k \in D'$
$\bar{t}_k \leftarrow \max\{\underline{t}_k, \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\}\}$, for $k \in D'$
$\underline{b}_k \leftarrow \min\{B - c_{j,k}, b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}\}$, for $k \in D'$
$\bar{b}_k \leftarrow \min\{B - c_{j,k}, b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}\}$, for $k \in D'$
sort depots in ascending order by $\underline{t}_k$, if tie decreasing by $\underline{b}_k$, if still tie decreasing by $\bar{b}_k$
rename depots according to new order from 1 to d'
if $\underline{t}_1 > u_j$ **then**
  Return: extension not possible
end
built the time-battery curve at customer j
$(\underline{t}_j^1, \underline{b}_j^1) \leftarrow (\underline{t}_1, \underline{b}_1), (\bar{t}_j^1, \bar{b}_j^1) \leftarrow (\bar{t}_1, \bar{b}_1)$
$b \leftarrow \bar{b}_1, f \leftarrow 2$
for $k = 2$ *to* d' **do**
  if $\underline{t}_k \leq u_j$ *and* $\bar{b}_k > b$ **then**
    $(\underline{t}_j^f, \underline{b}_j^f) \leftarrow (\underline{t}_k + \frac{1}{\alpha}(b - \underline{b}_k), b)$
    $(\bar{t}_j^f, \bar{b}_j^f) \leftarrow (\bar{t}_k, \bar{b}_k)$
    $b \leftarrow \bar{b}_k$
    $f \leftarrow f + 1$
  end
end
Return: time-battery curve of customer j

In the last phase of Algorithm 2 (lines 34 to 44), the time-battery curve at customer j is calculated and returned. The first element of the curve is given by the lower and upper time-battery pair corresponding to the first depot in the order. Hereafter, the depots are considered according to the order. The two pairs of a depot are added to the curve if both arrival times are within the time window of j and the maximal arrival SoC exceeds the highest SoC of the previous curve element, see line 38. The starting time and SoC of a new curve element are adjusted such that the lowest SoC equals the highest SoC of the predecessor element. After considering every depot, the program has calculated a time-battery curve at j and returns it.

The concept of the time-battery curve is a key to proving the algorithm's correctness. Thus, we first prove that Algorithm 1 calculates a time-battery curve at each customer of a given sequence.

Lemma 1. *Let s denote a customer sequence for which Algorithm 1 returns feasible. It follows that Algorithm 1 and 2 calculate a time-battery curve at each customer.*

The proof of Lemma 1 consists of several Claims and is provided in Appendix A. The correctness of the Algorithm is stated in the following theorem.

Theorem 1. *Algorithm 1 answers the decision problem, whether a feasible solution to a given customer sequence s exists, in polynomial time.*

The proof of Theorem 1 is in the Appendix B. Based on Theorem 1, Algorithm 1 can solve the decision problem, whether a given customer sequence is feasible or not, in polynomial time. With the extension discussed in Appendix B, Algorithm 1 is able to return such a feasible solution. We conclude by discussing a generalization of the algorithms and an equivalent method that evaluates a sequence of customers from back to front.

Observation 2. *Algorithm 2 can be used to evaluate arbitrary customer pairs since the input consists of a set of depots, two customers, and a time-battery curve.*

Based on Observation 2, Algorithm 2 can be embedded in a dynamic program that tries to find an *optimal* customer sequence according to some measure.

With Algorithm 1, we are able to consider a customer sequence from the first to the last customer. Similarly, one could consider a given sequence from the last to the first customer and decides on the feasibility. Note that the vehicle still executes the sequence from the first to the last customer, just the evaluation is in reversed order. We give a brief overview of the adaptions that need to be done for the reversed consideration of a sequence.

The interpretation of the two resources, namely time and battery, needs to be adapted. An algorithm has to consider the resources in a mirrored manner, which reflects in a traveled distance leading to a decrease in time and an increase of the SoC. Moreover, recharging decreases time and decreases the SoC. Suppose we interpret the two resources from a forward perspective. In that case, an arrival time at a customer or a depot is the latest possible arrival time. The respective SoC is the minimum required SoC to execute the already planned sequence.

In the following, we describe the modifications on Algorithm 1 to consider a sequence back to front. In the initialization (line 1), we set the arrival time of the last customer in the sequence as late as possible, and the SoC equals the travel time from the customer to the closest depot. In line 2, it is now checked if the arrival time is smaller than the lower time window. Moreover, the consideration of customers in the loop is in reserved order, and the Algorithm 3 is called. This algorithm performs the backward extension between two customers. A pseudocode of this algorithm is provided in Appendix C, see Algorithm 3.

Comment 1. *With these slight modifications of Algorithms 1 and 2, a reversed consideration of a customer sequence is possible, and the decision problem defined in Theorem 1 remains satisfied.*

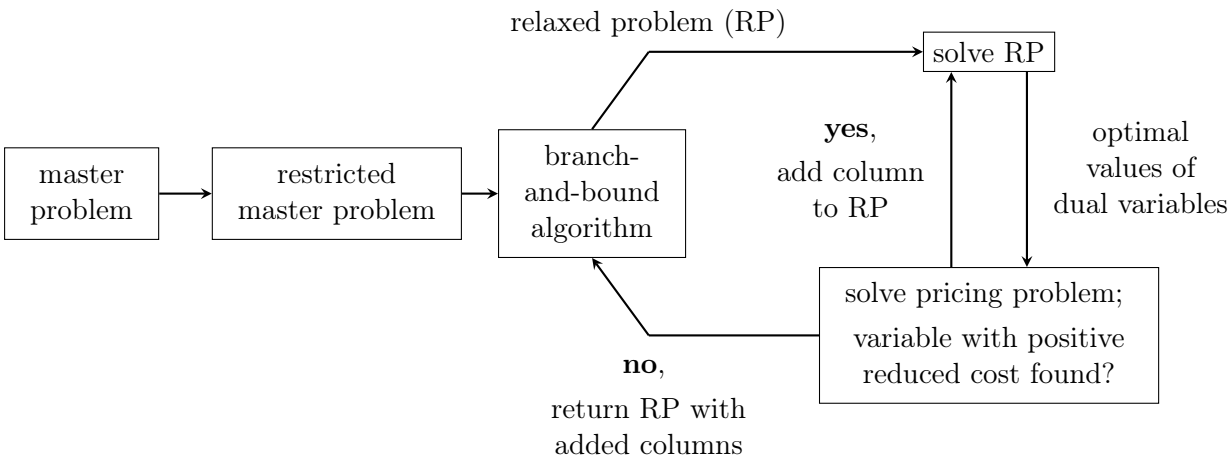


Figure 4: A schematic representation of a branch-and-price algorithm

Also the structure of the time-battery curves remains the same. However, its interpretation changes. In the backward case, time-battery pairs that a late and have a small SoC are favorable. This is the complete opposite of favorable time-battery pairs in the forward-oriented case. It can be shown that these modifications satisfy Theorem 1. In the following section, we provide a branch-and-price algorithm to solve the problem stated in Section 3 and explain how the forward- and backward-oriented dynamic program can be put together.

5 Branch-and-Price Algorithm

In this section, a branch-and-price algorithm to solve the problem stated in Section 3 is presented. A branch-and-price algorithm generally consists of a *branch-and-bound* approach, wherein each node of the tree is solved with column generation (see [6] and [14]). A schematic description of this approach is presented in Figure 4. For a good introduction into column generation, we refer to Desrosiers and Lübbecke [8].

We formulate our problem as a variant of the set-packing problem with an additional constraint. This formulation is called *master problem* (MP) in the following. Let Ω be the set of all feasible customer sequences. The formulation contains binary variables x_s, $s \in \Omega$, representing whether a feasible customer sequence s is performed by one of the vehicles. Note that Ω might contain exponentially many sequences in the number of customers, and calculating all those would be very time- and memory-consuming. Thus, we define a subset $\Omega' \subseteq \Omega$ and call the formulation, which only contains variables x_s, $s \in \Omega'$, *restricted master problem* (RMP). The RMP is given as the following Integer Program:

$$\text{maximize} \quad \sum_{s \in \Omega'} \sum_{i \in C} a_{is} p_i x_s \tag{1}$$

$$\text{subject to} \quad \sum_{s \in \Omega'} a_{is} x_s \leq 1 \qquad \forall\, i \in C \tag{2}$$

$$\sum_{s \in \Omega'} x_s \leq m \tag{3}$$

$$x_s \in \{0, 1\} \qquad \forall\, s \in \Omega' \tag{4}$$

Objective function (1) maximizes the total collected profit, whereas coefficient a_{is} is one if customer $i \in C$ is contained in sequence $s \in \Omega'$ (zero otherwise) and p_i is the profit of customer i. Inequalities (2) ensure that each customer $i \in C$ is served at most once, and constraint (3) guarantees that no more than m sequences/vehicles are selected. Finally, the domains of the variables are set by (4).

In each iteration, a linear relaxation of the RMP is solved, and the dual variables of an optimal solution are used to initialize the *pricing problem*. This problem identifies variables with positive reduced cost if any exist. These variables are then added to set Ω', and the linear relaxed RMP is solved again until no more variables with positive reduced cost exist. A detailed description of the pricing problem is given in the following subsection.

5.1 Pricing Problem

In order to solve the pricing problem, we define an auxiliary graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The set of nodes is denoted by $\mathcal{V} = FW \cup BW \cup \{v_0, v_{n+1}\}$, where $FW = \{i : i \in C, l_i \leq t^*\}$ and $BW = \{i : i \in C, u_i \geq t^*\}$ are the sets of forward and backward nodes, with $FW \cup BW = C$. Let v_0 and v_{n+1} denote the source and sink dummy-node. Value t^* is calculated, so that the size of sets FW and BW is balanced. The set of arcs is given as $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}\}$. Note that the set of depots is not contained in graph $\mathcal{G}$. Instead, a depot path between two customers is calculated by Algorithm 2. Let $P = (v_0, i_1, i_2, \ldots, i_j, v_{n+1})$ denote a path, where the first element is v_0, the last element is v_{n+1}, and the nodes in between represent a customer sequence. A path P is stated feasible if Algorithm 1 returns feasible for the customer sequence contained in path P. We define a trivial time window $[0, T]$ for v_0 and v_{n+1}, where T represents the sufficiently large time horizon. The travel time from customer $i \in C$ to the source or sink node is defined as c_i, which is the travel time to the nearest depot. Then, the pricing problem can be formulated as follows. Let ω_i, $i \in C$, be the optimal values of dual variables associated with constraints (2), and let ω_m denote the optimal value of the dual variable corresponding to constraint (3). Let $\bar{c_s}$, for $s \in \Omega$, be the reduced cost of variable x_s, i.e.

$$\bar{c}_s = \sum_{i \in C} a_{is}(p_i - \omega_i) - \omega_m. \tag{5}$$

An optimal solution of the pricing problem is given by $\bar{c}_s^* = \max_{s \in \Omega} \bar{c}_s$.

The pricing problem can be transformed into a variant of the elementary shortest-path problem with resource constraints, see Irnich and Desaulniers [12]. Since this problem is NP-hard, we propose a bidirectional labeling algorithm that is presented in the following.

5.2 Bidirectional Labeling Algorithm

The bidirectional labeling algorithm consists of a forward and backward labeling algorithm, whose results are combined in a union procedure. The forward labeling algorithm considers feasible partial-paths starting at node v_0 and ending at a node of $\mathcal{V}$. The backward labeling algorithm considers partial-paths in reversed order, meaning they start at node v_{n+1} and end at a node of BW. The size of the forward and backward sets is balanced to keep the computation times of the two algorithms similar.

Let $L = (i, X, \bar{c}, \underline{t}, \bar{t}, \underline{b}, \bar{b})$ denote a label, where $i \in \mathcal{V}$ is the current customer, set X contains all nodes that have been visited, $\bar{c}$ is the reduced cost of the label, and vectors $\underline{t}, \bar{t}, \underline{b}, \bar{b}$ represent the time-battery curve at i. A label can represents each state within both algorithms. The forward and backward labeling algorithms are initialized with labels $L_0^{fw} = (v_0, \{v_0\}, \omega_m, 0, 0, B, B)$ and $L_0^{bw} = (v_{n+1}, \{v_{n+1}\}, 0, T, T, 0, 0)$, respectively. Both algorithms repeatably extend each of their current labels to specific nodes. The forward labeling algorithm extends to feasible nodes

	forward label $L^{fw} = (i, X^{fw}, \bar{c}^{fw}, \underline{t}^{fw}, \bar{t}^{fw}, \underline{b}^{fw}, \bar{b}^{fw})$	backward label $L^{bw} = (i, X^{bw}, \bar{c}^{bw}, \underline{t}^{bw}, \bar{t}^{bw}, \underline{b}^{bw}, \bar{b}^{bw})$
feasibility rule a label can be extended to $j \in \mathcal{V}$ or $k \in BW$ if	$i \in FW, j \notin X^{fw}$, Algorithm 2 is successful	$k \notin X^{bw}$, Algorithm 3 is successful
extension rule a new label L'^{fw}/L'^{bw} is created according to the following rules	current node $= j$ $X'^{fw} = X^{fw} \cup j$ $\bar{c}'^{fw} = \bar{c}^{fw} + p_j - \omega_j$ $(\underline{t}'^{fw}, \bar{t}'^{fw}, \underline{b}'^{fw}, \bar{b}'^{fw}) =$ Algo.2 $(i, j, D, \underline{t}^{fw}, \bar{t}^{fw}, \underline{b}^{fw}, \bar{b}^{fw})$	current node $= k$ $X'^{bw} = X^{bw} \cup k$ $\bar{c}'^{bw} = \bar{c}^{bw} + p_k - \omega_k$ $(\underline{t}'^{bw}, \bar{t}'^{bw}, \underline{b}'^{bw}, \bar{b}'^{bw}) =$ Algo.3 $(i, k, D, \underline{t}^{bw}, \bar{t}^{bw}, \underline{b}^{bw}, \bar{b}^{bw})$
union rule label L^{fw} and L^{bw} at the same current node can be united if	$X^{fw} \cap X^{bw} = \{i\}$, $\exists t^{fw}, b^{fw}, t^{bw}, b^{bw}, l, k : t^{fw} \leq t^{bw}, b^{fw} \geq b^{bw}$, $\underline{t}_k^{fw} \leq t^{fw} \leq \bar{t}_k^{fw}, \underline{t}_\ell^{bw} \leq t^{bw} \leq \bar{t}_\ell^{bw}$, $b^{fw} = \underline{b}_k^{fw} + \alpha(t^{fw} - \underline{t}_k^{fw}), b^{bw} = \underline{b}_\ell^{bw} + \alpha(t^{bw} - \underline{t}_\ell^{bw})$	

Table 2: Feasibility, extension, and union rule for a forward and backward label.

of set $\mathcal{V}$ and the backward labeling algorithm to feasible nodes of set BW. The extension of a forward label is stopped if the current node is contained in $BW \setminus FW$ or is the sink node v_{n+1}.

All necessary steps of the bidirectional labeling algorithm, including a feasibility rule, an extension rule, and a union rule, are displayed in Table 2. The extension of a forward label to node $j \in \mathcal{V}$ is feasible if the current customer is contained in FW, node j is not visited yet, and the extension procedure (Algorithm 2) is successful. The extension of a backward label to a node $k \in BW$ is feasible if node k is not visited yet, and the backward extension procedure (Algorithm 3) is successful. A new label is created and stored in case of a successful extension. After both algorithms terminate, the union procedure combines all feasible forward and backward labels. The union of a forward and backward label is feasible if the following three conditions are satisfied:

(i) both labels are currently at the same node,

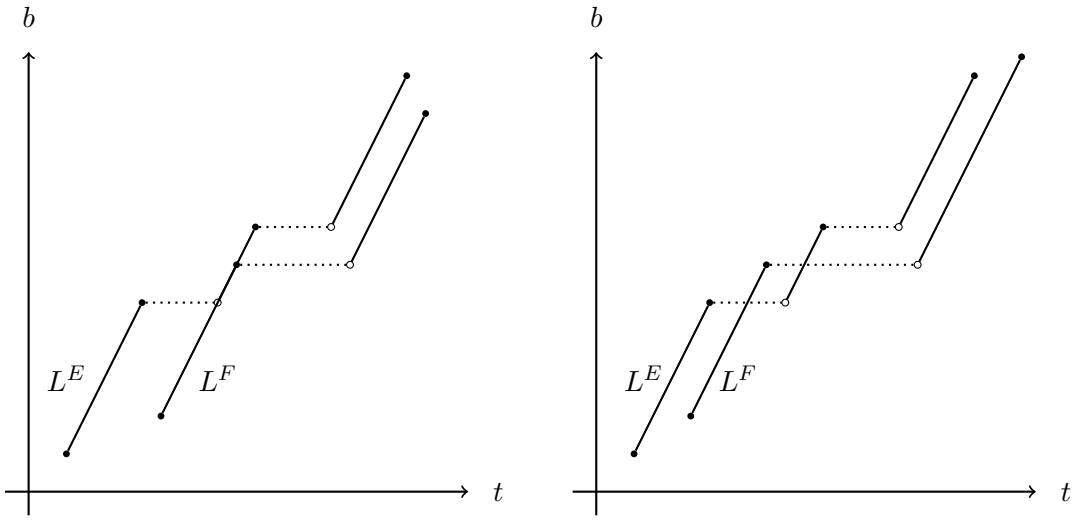
(ii) the only node visited by both labels is the current node, and

(iii) there is a time-battery pair from the time-battery curve of the forward label and one from the backward label's curve, such that the forward pair is not later and has a higher or equal SoC compared to the backward pair.

To speed up the bidirectional labeling algorithm, we use acceleration techniques that are described in the following.

5.3 Acceleration Techniques

The first technique is a dominance rule. The idea of dominance is to compare two labels at the same current customer, and check whether one can be stated better than the other. This check is based on the reduced cost, arrival time, SoC, and visited customers. In such a case, the labeling algorithm does not have to consider the weaker label further. Note that only forward labels are compared with forward labels and backward labels with backward labels. Let L^E and L^F be two labels that have the same current customer i. Moreover, let set Y contain all customers of $X^E \setminus X^F$ that are reachable from label L^F within their time window. Note that the reachability



(a) Time-battery curve of L^E dominates L^F.

(b) Two incomparable time-battery curves.

Figure 5: Two examples of the relation between time-battery curves.

check is kept simple, and all customers of Y are considered separately. We define $\bar{c}_Y$ as the sum of reduced cost of all customers contained in set Y. Let TB^E and TB^F denote the time-battery curve of L^E and L^F, respectively. We say that label L^E dominates label L^F if

(i) $\bar{c}_E \geq \bar{c}_F + \bar{c}_Y$,

(ii) $\forall\ (t^F, b^F) \in TB^F\ \exists\ (t^E, b^E) \in TB^E : t^E \leq t^F, b^E \geq b^F$ (forward), and

(ii) $\forall\ (t^F, b^F) \in TB^F\ \exists\ (t^E, b^E) \in TB^E : t^E \geq t^F, b^E \leq b^F$ (backward).

Condition (i) states that the reduced cost of L^E is greater or equal to the reduced cost of label L^F plus the reduced cost of set Y. Conditions (ii) and (iii) define the comparison of the time-battery curve of L^E and L^F for the forward algorithm and the backward algorithm, respectively. An example of the comparison of two time-battery curves is given in Figure 5. In Subfigure 5(a), the time-battery curve of label L^E dominates the time-battery curve of label L^F and in Subfigure 5(b), an example of two incomparable time-battery curves is displayed.

A further acceleration technique is based on reducing the number of nodes and arcs in graph $\mathcal{G}$. Instead of considering all nodes and arcs, we exclude arcs that lead to time or battery infeasible tours and nodes with negative reduced cost. An arc $(i, j) \in \mathcal{A}$ is removed from graph $\mathcal{G}$ if

$$l_i + \tau_i + \min_{k \in D}\{c_{i,k} + c_{j,k}\} + \frac{1}{\alpha}\max\{0, c_j + c_k + \min_{k \in D}\{c_{i,k} + c_{j,k}\} - B\} > u_j$$

holds because this inequality suggests that it is impossible to visit customer j immediately after customer i. This arc reduction is performed in a preprocessing phase of the branch-and-price algorithm. In addition to that, a possible reduction of the set of nodes is considered in each pricing iteration. As stated by equality (5), the reduced cost of a variable can be composed into customer components (minus value ω_m). This allows us to consider only customers with positive reduced cost, i.e. $p_i - \omega_i > 0$. These graph reductions can be applied to graph $\mathcal{G}$ without loss of optimality.

Apart from these generally valid graph reductions, we also apply a heuristic method to reduce the number of possible labels even further. For this purpose, we limit the number of outgoing arcs of each customer in $\mathcal{V}$ by a certain percentage and apply the bidirectional labeling algorithm on this reduced graph. To decide on the arcs to be discarded, we define a rating based on the following three components: (i) distance, (ii) difference between time windows, and (iii) reduced cost. The idea is to consider only the best-rated arcs. This heuristic speeds up the labeling procedure significantly. However, every time this heuristic cannot find variables with positive reduced cost, we have to apply the bidirectional labeling algorithm on the complete graph to guarantee optimality.

The last acceleration technique is a problem-specific branching rule. In the RMP, variables x_s can be either zero or one. Setting such a variable equal to one is very strong, whereas excluding a sequence $s \in \Omega'$ is fairly weak. Such a rule leads to a strong imbalance of the branch-and-bound tree, which increases the number of branching nodes and the number of pricing problems, which is generally time-consuming. To avoid this imbalance, we implement two branching rules. The first branching rule is called *node-branching*, where in one branch customer $i \in C$ is forced to be visited, see Constraint (6), and in the other branch the customer is discarded, see Constraint (7).

$$\sum_{s \in \Omega'} a_{is} x_s \geq 1 \tag{6}$$

$$\sum_{s \in \Omega'} a_{is} x_s \leq 0 \tag{7}$$

However, with this branching rule alone, the LP-solver can still calculate fractional solutions. To refine the branching, we introduce a second rule, called *arc-branching*. This rule enforces an arc $(i, j) \in \mathcal{A}$, with $i, j \in C$, to be either traversed, see constraint (8), or discarded, see constraint (9).

$$\sum_{s \in \Omega'} a_{(i,j)s} x_s \geq 1 \tag{8}$$

$$\sum_{s \in \Omega'} a_{(i,j)s} x_s \leq 0 \tag{9}$$

In the branching phase, we first check whether node-branching can be applied. If this is not the case, we then apply the arc-branching rule. Node-branching can be performed if the sum of variables visiting a specific customer is fractional. The rule selects the customer whose sum is closest to 0.5. Otherwise, we perform arc-branching on the arc whose sum is closest to 0.5. Note these additional branching constraints have to be respected when solving the pricing problem in the following way: Firstly, the additional constraint(s) should be respected in the reduced cost of each node. Secondly, branching constraints can exclude specific customers or arcs, which require simple modifications on the search graph $\mathcal{G}$.

6 Computational Studies

This section presents an extensive computational study on modified benchmark instances. We implemented the branch-and-price algorithm of Section 5 in the SCIP Optimization Suite 6.0.1 obtained by Zuse Institute Berlin [11]. To solve the linear relaxations, we used SoPlex 4.0.1. All experiments were performed on a single core of a computer with a 2.90 GHz CPU running Windows 10 as the operating system.

Since no instances for vehicles with a single-unit capacity and multiple depots exist, we decided to use benchmark instances for the multiple depot vehicle routing problem with time

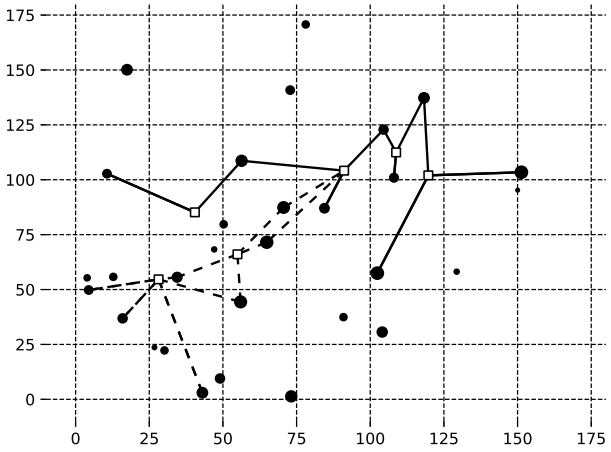


Figure 6: Solution of an instance with 30 customers, six depots, and two vehicles.

windows, as proposed by Cordeau et al. [5]. The set consists of 20 instances, which are available under http://bernabe.dorronsoro.es/vrp/. The instances are divided into two groups with different a distribution of the time windows. The first 10 instances have narrow time windows with a length of 90 to 180, starting between 60 and 480, and the second 10 instances have wider time windows with a length between 180 and 360, beginning between 60 and 300. The number of customers ranges between 48 and 288, and four or six depots are considered. We assumed that all depots function as parcel storage and recharging station. The battery capacity equaled 250, and the charging rate α was set to 2.5. Since the instances only contain customer and depot locations coordinates, we calculated the Euclidean distances between each pair of locations and rounded these values up to the next integer. Service durations are individual for each customer and contained in the instances. However, the instances do not contain any profits. We, therefore, used the original customer demand as profit.

Besides these instance-related parameters, we defined additional parameters for the branch-and-price algorithm. Firstly, the number of variables that are added in each pricing problem. Adding only the single variable with the highest reduced cost keeps the RMP small but it could result in many pricing problems. On the other hand, adding all variables with positive reduced cost increases the size of the RMP significantly but could reduce the number of pricing problems. In a testing phase, we observed that bounding the number of added variables per pricing problem by 500 showed the best results. Furthermore, we define the percentage of outgoing arcs per customer in the heuristic pricing method. This parameter defines the density of the graph. A sparse graph contains fewer paths, and the pricing problem is solved quickly. However, this heuristic might fail to find promising variables, and the complete graph needs to be considered. Tests suggested that keeping 30 % of all outgoing arcs leads to good results. We fixed these parameters for all following experiments.

An example of an optimal solution for an instance with 30 customers (circles), six depots(squares), and two vehicles is displayed in Figure 6. The size of the black circles corresponds to the profit collected when served. The tours of the two vehicles are represented by the solid and dashed lines, respectively. Note that if a customer is connected by a single line to a depot, the vehicle travels from the depot to the customer and back to the same depot. Customers that are isolated are not served. One can see that profits and distances affect the customer selection. Furthermore, the vehicles do not only operate from a single depot but change it several times.

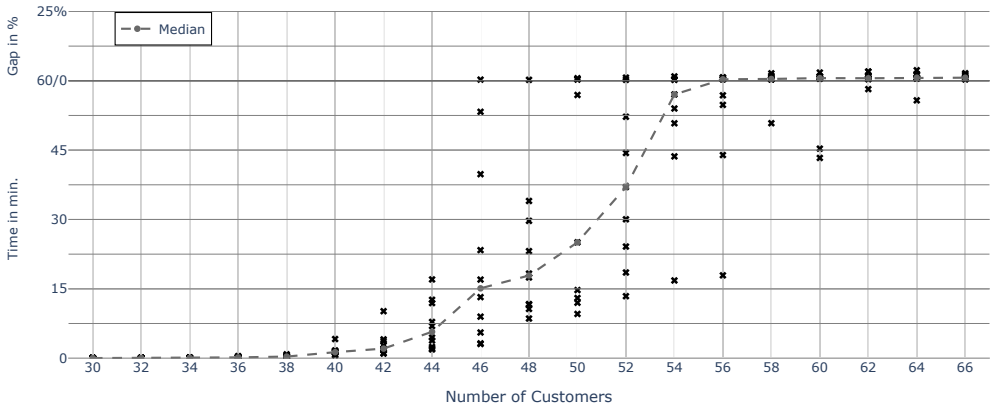


Figure 7: Computing times for instances with 30 to 66 customers.

6.1 Computational Results

We designed three experiments to analyze the influence of varying customers, vehicles, and time window lengths on the computation time of our algorithm. Furthermore, we provide detailed results for the previously described benchmark instances. For each of the computations, we set a time limit of one hour. We report the respective gap for those instances that are not solved within the time limit. To relate the gap to the solution time, we split the y-axis into two parts in each of the following plots: a lower section that shows the solution time in minutes and an upper section with the respective gap in percent for those instances that timed out. We created ten samples per instance in all three experiments by randomly selecting customers to generate more reliable results. For clarity, we calculated a value for each sample according to the formula: solution time + gap $\cdot$ 3,600, and only report the median of the ten samples for each instance. To visualize a trend, we added a median line of the respective plotted values.

In the first experiment, we investigated the influence of a varying customer number on the computing time. For this purpose, we fixed the number of vehicles to five, and all other parameters are set as previously described. The number of customers ranged from 30 to 66 with a step size of two. To generate more comparable results, we only considered the first 10 instances, where time window lengths are similar. Note that the first instance of the benchmark set only contains 48 customers and is, therefore, discarded for all tests, where this customer number is exceeded.

Results of this experiment are presented in Figure 7. It is visible that instances with up to 40 customers can be solved in a short time. Furthermore, we found out that for instances with 40 customers, the ratio between the collected profit and the total collectible profit is 97% on average, which is one possible explanation for this solution time. For instances with more than 40 customers, the computing time strongly increases, and already with 46 customers, the median of one instance is not within the time limit. For those instances, with 56 customers, nearly half of the reported medians are not within the time limit, and for 66 customers, none of the medians are within the time limit. The dashed gray median line visualizes this trend in the chart. Nevertheless, the reported gaps are small, with up to 5%. We can infer that the solvability strongly depends on the number of customers. An explanation for this is that a larger search graph significantly increases the computing time of the bidirectional labeling algorithm.

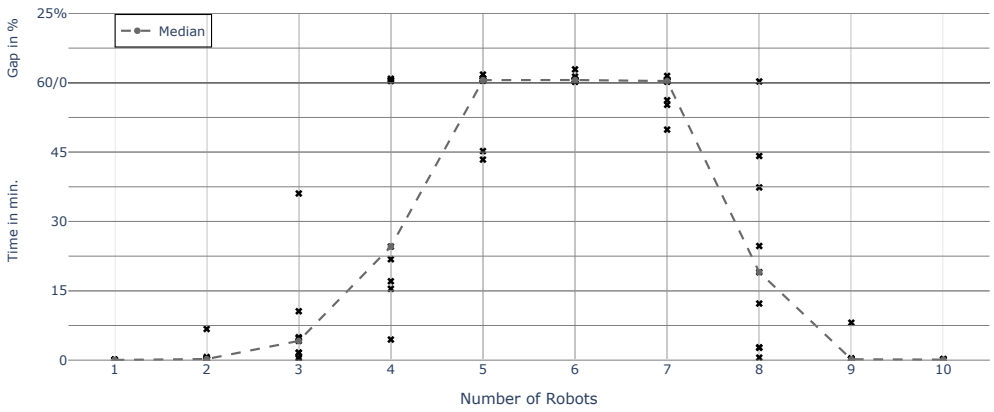


Figure 8: Computing times for instances with 1 to 10 vehicles.

The second experiment evaluates the influence of varying the number of vehicles on the computation time. For this purpose, we fixed the number of customers to 60, and all other parameters are chosen as previously. Similar to the last experiment, we considered the first 10 instances, of which the first instance only contains 48 customers and is therefore discarded. The results of this experiment are displayed in Figure 8. It turned out that instances with one, two, nine, and ten vehicles can be solved quickly, whereas instances with five to seven vehicles are either unsolved or the computing time is close to the time limit. In the case of of six vehicles, the median of all instances is above the time limit, even though the gaps remain small.

This experiment shows that the computing times not only depend on the number of customers but also on the ratio between customers and vehicles. If vehicles are not a rare resource, our algorithm calculates an optimal solution fast. This observation also holds for a small number of vehicles. A possible explanation is that, in the case of many vehicles, almost all customers are served, and vehicles might not be fully utilized. In the case of a small number of vehicles, it seems relatively easy to assign the most profitable tours to vehicles, and inferences between tours do not occur as often.

In a third experiment, we investigated the influence of the time window length on the computing time. We varied the time window length between 15 and 45. The new time windows are calculated as follows: the upper time window bounds remained unchanged, and the lower bounds were adjusted by subtracting the time window length from the original upper bounds. We considered 100 customers per instance and chose eight vehicles in order to maintain a good ratio. All other values are fixed and set as before. In this experiment, all customers have time windows of equal length. Therefore, we considered the 14 instances with more than 100 customers. The effect of varying the time window length can be seen in Figure 9. It is visible that wider time windows result in larger computing times. The median curve shows that for a time window length of 45, most instances cannot be solved within one hour. This effect is because wider time windows increase the number of arcs in the search graph, which affects the number of labels in the bidirectional labeling algorithm and, thus, the computing time.

In the last experiment, we tested our algorithm on the original MDVRPTW instances with reduced time window lengths of 15 units. The amount of vehicles for each instance equals the number of customers divided by 12, and the result is rounded down to the next integer value. All other parameters are equal to the previous experiments. The results are presented in Table 3.

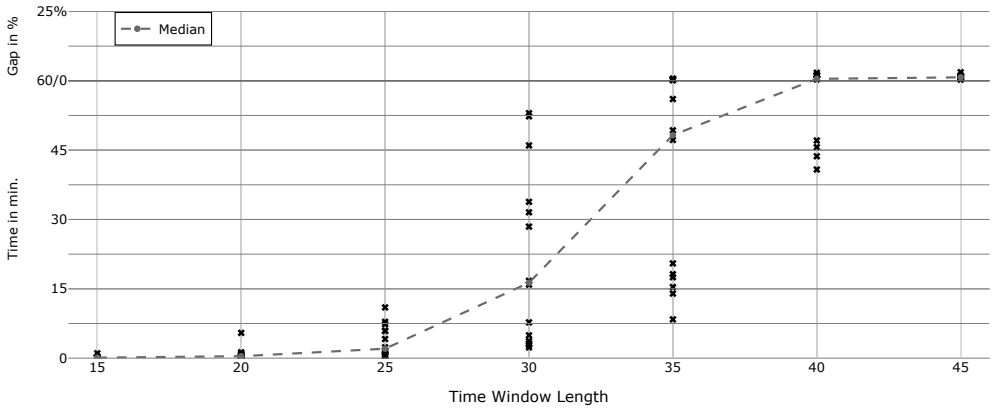


Figure 9: Computing times for instances with time window lengths of 15 to 45.

The first four columns specify the instance parameters, and the best calculated lower bound is displayed in the fifth column. Columns six and seven show the solution time in seconds and the number of nodes in the branch-and-price tree. Lastly, for those instances that have not been solved within the time limit, the gaps (in percent) are reported. With this parameter setting, we are able to solve all instances with up to 144 customers and even one instance with 192 customers within one hour. Moreover, an optimal solution of instances with less than 100 customers can be found by considering only a few nodes of the branch-and-price tree, which might be reasoned by our individual branching rules. However, the number of nodes grows strongly in the number of customers as can be seen for example for instances 4, 14 and 19. An exception is instance 18, which is already solved within the root node. Note that for those instances that timed out, the actual number of branch-and-price nodes required to solve the problem to optimality might be higher.

7 Conclusion

In this work, we presented a branch-and-price algorithm to solve a problem motivated by innovative delivery concepts for the last-mile with multiple micro-depots. Our problem is related to the team orienteering problem with time windows and recharging stations with the difference that the vehicles considered in this work have a single-unit capacity. We presented a dynamic program, which can solve the following decision problem for a given customer sequence. Is there a feasible path of depots between each pair of customers such that the battery remains within the bounds and customers' time windows are satisfied? This dynamic program is used to solve the pricing problem within the branch-and-price algorithm. An essential aspect of our work and demarcation from previous publications is that the dynamic program allows us to consider the recharging stations implicitly in the search graph of the pricing problem. This fact reduces the size of the search graph in the labeling algorithm and allows multiple depot visits between two customers. The results of our computational studies show that the branch-and-price algorithm is able to solve instances of reasonable size in short time. Also, the experiments show that the computation time depends on the number of customers, the ratio between customers and vehicles, and the time window length. Moreover, for small time windows of 15 units, the algorithm

instance	cust	depots	veh	LB	time(s)	BnP-nodes	gap
1	48	4	4	453	0.65	1	-
2	96	4	8	928	5.93	2	-
3	144	4	12	1321	81.09	43	-
4	192	4	16	1928	3600	366	0.12 %
5	240	4	20	2470	3600	9	7.42 %
6	288	4	24	2581	3600	3	18.03%
7	72	6	6	672	2.14	4	-
8	144	6	12	1479	54.38	7	-
9	216	6	18	1998	3600	41	15.33%
10	288	6	24	2999	3600	24	3.06 %
11	48	4	4	398	0.33	1	-
12	96	4	8	787	3.6	7	-
13	144	4	12	1226	162.8	82	-
14	192	4	16	1631	1102.63	195	-
15	240	4	20	2241	3600	156	3.34 %
16	288	4	24	2690	3600	59	1.15 %
17	72	6	6	598	1.246	3	-
18	144	6	12	1357	15.38	1	-
19	216	6	18	2013	3600	332	0.67 %
20	288	6	24	2565	3600	57	8.67 %

Table 3: Results for the MDVRPTW instances with a time window length of 15

solves instances of up to 144 customers in under one hour.

The vehicles have a single-unit capacity in our current model, which simplifies the model. For future work, we propose extending this model and increasing the vehicle's capacity. Furthermore, it would be interesting to investigate how the dynamic program can be modified to be suitable for the EVRP and to which extent it would speed up the pricing problem of a branch-and-price algorithm. In addition, one could consider a more realistic recharging and consumption model and check how our branch-and-price algorithm can be modified towards such a model.

Another research direction is integrating different speed levels into the model, influencing the battery consumption. Especially for drones, which fly at high speeds (up to 120 km/h), the selection of speeds could impact consumption significantly. Therefore, speed optimization would be beneficial. Similar to the solution approach of our pricing problem, the idea is to find a dynamic program that calculates optimal recharging times and speed levels for a given customer sequence.

Appendix A: Proof of Lemma 1

In the following, we prove that Algorithms 1 and 2 calculate a time-battery curve, as defined in Definition 2, at each customer of a given sequence if the sequence is stated feasible. The first claim shows that the lower arrival time $\underline{t}_k$ is never later than the upper arrival time $\bar{t}_k$ at a customer from a depot k and that the same relation holds for the corresponding lower and upper SoC.

We assume that the service time τ equals zero for each customer throughout this section. This assumption is without loss of generality: starting with the first customer, we put τ to zero and reduce the time window of all subsequent customers by τ. Moreover, let all references to algorithmic lines refer to Algorithm 2. If a line reference refers to Algorithm 1, it is explicitly mentioned.

Claim 1. *Let $k \in D$ be a depot, which is visited at (t_k, b_k), and let $j \in C$ be a customer. Algorithm 2 calculates a lower and upper arrival time $\underline{t}_k$ and $\bar{t}_k$, and a lower and upper SoC $\underline{b}_k$ and $\bar{b}_k$ from k at j. It follows that $\underline{t}_k \leq \bar{t}_k$ and $\underline{b}_k \leq \bar{b}_k$ holds.*

Proof. The calculation of lower and upper values is given in lines 26 to 29. In the following, we assume that $\underline{t}_k \leq u_j$ and $c_{j,k} + c_j \leq B$ holds, otherwise the two values are excluded from consideration, see lines 23, 32, and 38.

We first show that $\underline{t}_k \leq \bar{t}_k$ holds. The upper arrival time $\bar{t}_k$ is calculated as

$$\max\{\underline{t}_k, \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\}\},$$

see line 27. Based on the assumption that $\underline{t}_k$ is less or equal to the upper time window bound of j, this inequality holds.

It remains to show that $\underline{b}_k \leq \bar{b}_k$ holds. The lower SoC $\underline{b}_k$ is calculated as

$$\min\{B - c_{j,k}, b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}\},$$

see line 28, and the upper SoC $\bar{b}_k$ is calculated as

$$\min\{B - c_{j,k}, b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}\},$$

see line 29. Thus, this inequality holds, since $\underline{t}_k$ is always less or equal to the upper time window bound of j.

□

In the subsequent claim, we show that the lower arrival time and the corresponding SoC, calculated by Algorithm 2 at a customer, form a time-battery pair. The same holds for the upper arrival time and SoC. The Definition of time-battery pairs is given in the following.

Definition 3. *Let k and j, with $k, j \in V$, be two nodes. Node k is visited at t_k with an SoC of b_k and from k node j is visited at t_j with an SoC of b_j. The pair (t_j, b_j) is called time-battery pair if*

$$t_j \geq t_k + r_k + c_{k,j}$$
$$b_j \leq b_k + \alpha r_k - c_{k,j}$$

holds.

Claim 2. *Let $j \in C$ be a customer, $k \in D$ a depot, let t_k be the arrival time, and b_k the SoC at k. It holds that Algorithm 2 calculates a lower time-battery pair $(\underline{t}_k, \underline{b}_k)$ and an upper time-battery $(\bar{t}_k, \bar{b}_k)$ at j or removes depot k from consideration.*

Proof. We first show that lower and upper time-battery pair, calculated by Algorithm 2, are either within the customer's time window or not considered. The calculated arrival times are feasible if $[\underline{t}_k, \bar{t}_k] \subseteq [l_j, u_j]$ holds and the SoCs are feasible if $[\underline{b}_k, \bar{b}_k] \subseteq [c_j, B - c_{j,k}]$ holds. Both arrival times are at least l_j since $\underline{t}_k$ is at least l_j, see line 26, and Claim 1 shows that $\underline{t}_k \leq \bar{t}_k$. Moreover, both arrival times are equal or lower than u_j or removed from consideration, see lines 32 and 38. It remains to show that $[\underline{b}_k, \bar{b}_k] \subseteq [c_j, B - c_{j,k}]$ holds. The interval $[c_j, B - c_{j,k}]$ is not empty since $c_{j,k} + c_j < B$, see line 22. The lower SoC $\underline{b}_k$ is at least c_j, which is ensured by the definition of the recharging time in the calculation of $\underline{t}_k$, see line 26. Based on Claim 1, it holds that $\underline{b}_k \leq \bar{b}_k$. Moreover, $\underline{b}_k$ and $\bar{b}_k$ are both bounded by $B - c_{j,k}$, see line 28 and 29. Thus, $[\underline{b}_k, \bar{b}_k] \subseteq [c_j, B - c_{j,k}]$ holds.

Next, we prove that both values are obtainable from depot k. The definition of the lower arrival time $\underline{t}_k$ is

$$\max\{l_j, t_k + \frac{1}{\alpha} \max\{0, c_{j,k} + c_j - b_k\} + c_{j,k}\},$$

see line 26. In general, the earliest arrival time at j is given as $t_k + r_k + c_{j,k}$, where r_k correspond to the minimal required recharging time at k. This time is either zero or given as $\frac{1}{\alpha}(c_{j,k} + c_j - b_k)$. A second lower bound to the arrival time is l_j. Both conditions together represent the definition of $\underline{t}_k$. The definition of the lower SoC $\underline{b}_k$ is

$$\min\{b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}, B - c_{j,k}\},$$

see line 28. In general, the SoC upon arrival at j is determined as $b_k + \alpha r_k - c_{j,k}$, where the recharging time can be expressed as $\underline{t}_k - t_k - c_{j,k}$. However, the maximal SoC at j from k is $B - c_{j,k}$. Both conditions together represent the definition of $\underline{b}_k$. Thus, $\underline{t}_k$ and $\underline{b}_k$ are obtainable from depot k. It remains to show that $\underline{t}_k$ and $\underline{b}_k$ form a time-battery pair as defined in Definition 3. This means that a recharging time r_k at depot k exists such that (i) $\underline{t}_k \geq t_k + r_k + c_{j,k}$ and (ii) $\underline{b}_k \leq b_k + \alpha r_k - c_{j,k}$. Choose $r_k = \frac{1}{\alpha} \max\{0, c_{j,k} + c_j - b_k\}$. It follows that

$$t_k + r_k + c_{j,k} = t_k + \frac{1}{\alpha} \max\{0, c_{j,k} + c_j - b_k\} + c_{j,k} \underset{\text{(line 26)}}{\leq} \underline{t}_k.$$

Thus, inequality (i) is fulfilled. For the SoC at j it follows,

$$b_k + \alpha r_k - c_{j,k} \overset{\text{(i)}}{\geq} b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k} \geq \min\{B - c_{j,k}, b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}\} \underset{\text{(line 28)}}{=} \underline{b}_k.$$

Thus, $(\underline{t}_k, \underline{b}_k)$ form a time-battery pair.

Next, we prove that the upper arrival time $\bar{t}_k$ and the upper SoC $\bar{b}_k$ are also obtainable from depot k. The upper arrival time $\bar{t}_k$ is defined as

$$\max\{\underline{t}_k, \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\}\},$$

see line 27. Based on Claim 1, it holds that $\underline{t}_k \leq \bar{t}_k$. Moreover, customer j is reachable at $\underline{t}_k$ and $\bar{t}_k$ is within the time window of j, which means $\bar{t}_k \leq u_j$. Thus, $\bar{t}_k$ is obtainable. The upper SoC is defined as

$$\bar{b}_k = \min\{b_k + \alpha(u_k - t_k - c_{j,k}) - c_{j,k}, B - c_{j,k}\},$$

see line 29. The first part of the term represents the SoC at j if the vehicle arrives on the upper time window bound of j. The additional time is used to recharge. The second part corresponds to the SoC at j after a full recharge at k. Since $\bar{b}_k$ is defined as the minimum of both values, the upper SoC is obtainable from k. It remains to show that upper time and the upper SoC form a

time-battery pair. This means that a recharging time r_k exists, such that (iii) $\bar{t}_k \geq t_k + r_k + c_{j,k}$ and (iv) $\bar{b}_k \leq b_k + \alpha r_k - c_{j,k}$ holds. We choose $r_k = \min\{\frac{1}{\alpha}(B - b_k), u_j - t_k - c_{j,k}\}$. It follows that

$$\begin{aligned} t_k + r_k + c_{j,k} &= t_k + \min\{\frac{1}{\alpha}(B - b_k), u_j - t_k - c_{j,k}\} + c_{j,k} \\ &= \min\{t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}, u_j\} \\ &\leq \max\{\underline{t}_k, \min\{t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}, u_j\}\} \\ &\underset{\text{(line 27)}}{=} \bar{t}_k. \end{aligned}$$

For the upper SoC at j it follows,

$$\begin{aligned} b_k + \alpha r_k - c_{j,k} &= b_k + \alpha \min\{\frac{1}{\alpha}(B - b_k), u_j - t_k - c_{j,k}\} - c_{j,k} \\ &= \min\{B - c_{j,k}, b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}\} \\ &\underset{\text{(line 29)}}{=} \bar{b}_j. \end{aligned}$$

Thus, the upper arrival time and the SoC form a time-battery pair.

□

Claim 3. *Let $\in D$ be a depot, which is visited at (t_k, b_k), and $j \in C$ a customer. Algorithm 2 calculates a lower time-battery pair $(\underline{t}_k, \underline{b}_k)$ and an upper time-battery pair $(\bar{t}_k, \bar{b}_k)$ from k at j. It holds $\underline{t}_k = \bar{t}_k$ if and only if $\underline{b}_k = \bar{b}_k$.*

Proof. Equally to Claim 1, we assume that both pairs are feasible. Suppose $\underline{t}_k = \bar{t}_k$ holds. Based on line 27, it follows that $\underline{t}_k \geq \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\}$. We distinguish the two following cases: (i) $\underline{t}_k \geq u_j$ or (ii) $\underline{t}_k \geq t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}$. In the first case, it follows directly from the definition of $\underline{b}_k$ and $\bar{b}_k$ that $\bar{b}_k = \underline{b}_k$ holds, see line 28 and 29. In the second case, it holds that $\underline{t}_k \geq t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}$, and it follows

$$\begin{aligned} \underline{b}_k &\underset{\text{(line 28)}}{=} \min\{B - c_{j,k}, b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}\} \\ &\geq \min\{B - c_{j,k}, b_k + \alpha(t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k} - t_k - c_{j,k}) - c_{j,k}\} \\ &= \min\{B - c_{j,k}, B - c_{j,k}\} \\ &= B - c_{j,k} \\ &\underset{\text{(line 29)}}{\geq} \bar{b}_k. \end{aligned}$$

Together with Claim 1, it follows that $\underline{b}_k = \bar{b}_k$ holds.

Suppose $\underline{b}_k = \bar{b}_k$ holds. Following lines 28 and 29, it suffices to distinguish the two following cases: (i) $\underline{b}_k = \bar{b}_k = B - c_{j,k}$ or (ii) $\underline{b}_k = b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}$ and $\bar{b}_k = b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}$. In the first case, it follows

$$\begin{aligned} & B - c_{j,k} \leq b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k} \\ \Leftrightarrow\ & \underline{t}_k \geq t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}. \end{aligned}$$

The upper arrival time from depot k is defined as

$$\bar{t}_k \underset{\text{(line 27)}}{=} \max\{\underline{t}_k, \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\}\}.$$

Suppose the maximum is given by the second term, otherwise $\bar{t}_k = \underline{t}_k$ follows directly. It follows that

$$\bar{t}_k = \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\} \leq t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k} \leq \underline{t}_k.$$

Claim 1 implies $\underline{t}_k = \bar{t}_k$.

It remains to show that the second case also implies that the lower and upper arrival times are equal. In the second case, where $\underline{b}_k = b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}$ and $\bar{b}_k = b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}$ holds, it follows that $\underline{t}_k = u_j$. This implies that

$$\bar{t}_k \underset{\text{(line 27)}}{=} \max\{\underline{t}_k, \min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\}\} = u_j = \underline{t}_k.$$

□

The following claim shows that the slope of the linear interpolation between a lower and upper time-battery pair, calculated by Algorithm 2, is equal to the recharging rate α.

Claim 4. *Let $k \in D$ be a depot, which is visited at (t_k, b_k), and let $j \in C$ be a customer. Algorithm 2 calculates a lower time-battery pair $(\underline{t}_k, \underline{b}_k)$ and an upper time-battery pair $(\bar{t}_k, \bar{b}_k)$ from k at j. If $\underline{t}_k \neq \bar{t}_k$, the slope of the linear interpolation between these two pairs is the recharging rate α.*

Proof. Equally to Claim 1, we assume that both pairs are feasible. Since $\underline{t}_k \neq \bar{t}_k$, it follows based on Claim 3 that $\underline{b}_k \neq \bar{b}_k$. The slope of the linear interpolation can be calculated as:

$$\frac{\bar{b}_k - \underline{b}_k}{\bar{t}_k - \underline{t}_k} \underset{\text{line } 26-29}{=} \frac{\min\{B - c_{j,k}, b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}\} - \min\{B - c_{j,k}, (b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k})\}}{\min\{u_j, t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}\} - \underline{t}_k}$$

Based on Claim 1 and 3, it holds that $\underline{b}_k = b_k + \alpha(\underline{t}_k - t_k - c_{j,k}) - c_{j,k}$. We arrive with a distinction in four cases:

(i) $\bar{b}_k = b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}$ and $\bar{t}_k = u_j$,

(ii) $\bar{b}_k = b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}$ and $\bar{t}_k = t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}$,

(iii) $\bar{b}_k = B - c_{j,k}$ and $\bar{t}_k = t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}$, and

(iv) $\bar{b}_k = B - c_{j,k}$ and $\bar{t}_k = u_j$.

In case (i), it follows that

$$\frac{b_k + \alpha(u_j - t_k - c_{j,k}) - b_k - \alpha(\underline{t}_k - t_k - c_{j,k})}{u_j - \underline{t}_k} = \alpha.$$

Case (ii): It follows that $\bar{b}_k \leq B - c_{j,k}$, which is equivalent to

$$\alpha(u_j - t_k - c_{j,k}) \leq B - b_k.$$

It follows $\bar{t}_k = u_j$, since $\bar{t}_k \geq t_k + \frac{1}{\alpha}(\alpha(u_j - t_k - c_{j,k})) + c_{j,k} = u_j$ and $\bar{t}_k$ is feasible ($\bar{t}_k \leq u_j$). Thus, we are in case (i).

In case (iii), it follows that

$$\frac{B - b_k - \alpha(\underline{t}_k - t_k - c_{j,k})}{t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k} - \underline{t}_k} = \alpha.$$

Case (iv): It follows that $\bar{b}_k \leq b_k + \alpha(u_j - t_k - c_{j,k}) - c_{j,k}$, which is equivalent to

$$u_j \geq t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}.$$

Thus, $\bar{t}_k = u_j = t_k + \frac{1}{\alpha}(B - b_k) + c_{j,k}$ and we are in case (iii).

□

It remains to show that all pairs on the linear interpolation between the lower and upper pair are feasible and obtainable.

Claim 5. *Let $k \in D$ be a depot, which is visited at time t_k with an SoC b_k, and let $j \in C$ be a customer. Algorithm 2 calculates a feasible lower time-battery pair $(\underline{t}_k, \underline{b}_k)$ and a feasible upper time-battery pair $(\bar{t}_k, \bar{b}_k)$ from k at j. It holds that all pairs on the linear interpolation between the lower and upper pair are obtainable from k.*

Proof. In Claim 2 it was shown that the lower and upper time-battery pair are feasible. This means that the lower and upper pair at j is reachable. Claim 4 states that the slope of the linear interpolation is α. Thus, every point on the interpolation can be obtained by recharging since the SoC increases by α for each time unit charged. This holds until the battery capacity B is reached, which corresponds to the upper pair on the interpolation if a full recharge is possible concerning the time window of j.

□

The next claim shows that the calculations of Algorithm 2 satisfies the first condition of a time-battery curve, see Definition 2.

Claim 6. *Let i and $j \in C$ be two sequential customers and let $\underline{p}_1 = (\underline{t}_1, \underline{b}_1)$ and $\underline{p}_2 = (\underline{t}_2, \underline{b}_2)$ denote two lower time-battery pairs at customer j calculated by Algorithm 2. If $\underline{t}_1 < \underline{t}_2$ holds, it follows that $\underline{b}_1 \geq \underline{b}_2$.*

Proof. We assume that a time-battery curve is given at customer $i \in C$. Proof by contradiction: Suppose $\underline{b}_1 < \underline{b}_2$ holds. Let s_1 denote the depot sequence between i and j in order to arrive at the lower time-battery pair $\underline{p}_1$ and let s_2 denote the depot sequence in order to arrive at $\underline{p}_2$. Value $\underline{t}_2$ is greater than l_j, since $\underline{t}_1 < \underline{t}_2$ holds. Moreover, the depot closest to j is at most $\underline{b}_1$ units away, since the lower SoC is calculated such that the nearest depot is reachable, see line 26. Based on the assumption $\underline{b}_1 < \underline{b}_2$, it follows that no recharge is conducted on s_2 in order to arrive at $\underline{p}_2$, since Algorithm 2 keeps the recharging time minimal, see line 15 and 26. Moreover, it selects the first time-battery pair at i which has an SoC that is sufficient to reach the first depot on s_2, line 5 and 6. Thus, the lowest time-battery pair on the curve at i is already sufficient to cover s_2 and the vehicle arrives at $\underline{p}_2$. If the earliest pair is selected and $\underline{t}_1 < \underline{t}_2$ holds, it follows that the length of s_2 is greater than s_1. However, this contradicts the assumption that $\underline{b}_1 < \underline{b}_2$ holds, since no recharge is conducted on s_2 in order to arrive at $\underline{p}_2$, the vehicle starts earliest, and the length s_2 exceeds that of s_1.

□

Proof. Proof of Lemma 1: We provide a proof by mathematical induction.

Base case: First, we have to prove that Algorithm 1 calculates a time-battery curve at the first customer. In line 1 of Algorithm 1, the time-battery curve at the first customer is calculated. It consists of a single time-battery pair calculated from the depot closest to the first customer, which is by definition a time-battery curve.

Inductive Step: Let i, j denote two sequential customers of sequence s. Moreover, we are given a time-battery curve at i. Algorithm 2 calculates time-battery pairs in the extension from

i to j from each depot of set D', see lines 26 to 29. We show that Algorithm 2 calculates a time-battery curve at j.

The definition of a time-battery curve is given in Definition 2. Lower time-battery pairs $(\underline{t}_j^1, \underline{b}_j^1), (\underline{t}_j^2, \underline{b}_j^2), \ldots, (\underline{t}_j^q, \underline{b}_j^q)$ and upper time-battery pairs $(\bar{t}_j^1, \bar{b}_j^1), (\bar{t}_j^2, \bar{b}_j^2), \ldots, (\bar{t}_j^q, \bar{b}_j^q)$ are calculated in lines 39 to 40. In the following, we prove that all four conditions of a time-battery curve are fulfilled:

(i) $\underline{t}_j^f \leq \bar{t}_j^f$ and $\underline{b}_j^f \leq \bar{b}_j^f$ for all $f = 1, \ldots, q$,

(ii) $\bar{t}_j^f < \underline{t}_j^{f+1}$ and $\bar{b}_j^f = \underline{b}_j^{f+1}$ for all $f = 1, \ldots, q-1$,

(iii) $l_j \leq \underline{t}_j^1$ and $\bar{t}_j^q \leq u_j$, and

(iv) $c_j \leq \underline{b}_j^1$ and $\bar{b}_j^q \leq B - c_j$.

(i) Let $k \in D$ denote the depot from which customer j is visited on curve element f of the time-battery curve. Moreover, let $(\underline{t}_k, \underline{b}_k)$ denote the lower time-battery pair and $(\bar{t}_k, \bar{b}_k)$ the upper time-battery pair from k at j, see lines 26 to 29. It follows that

$$\underline{t}_j^f \underset{\text{(line 39)}}{=} \underline{t}_k + \frac{1}{\alpha}(b - \underline{b}_k) \underset{\text{(line 38)}}{<} \underline{t}_k + \frac{1}{\alpha}(\bar{b}_k - \underline{b}_k) \overset{\text{(Claim 4)}}{=} \bar{t}_k \underset{\text{(line 40)}}{=} \bar{t}_j^f.$$

For the SoC, it follows that

$$\underline{b}_j^f \underset{\text{(line 39)}}{=} b \underset{\text{(line 38)}}{<} \bar{b}_k \underset{\text{(line 40)}}{=} \bar{b}_j^f.$$

Thus, condition (i) is fulfilled.

(ii) We choose depot $k \in D$ as in case (i). Let $k' \in D$ denote the depot from which customer j is visited on curve element $f+1$ of the time-battery curve. Moreover, let $(\underline{t}_{k'}, \underline{b}_{k'})$ denote the lower time-battery pair and $(\bar{t}_{k'}, \bar{t}_{k'})$ the upper time-battery pair calculated from k' at j. We first prove that $\bar{b}_j^f = \underline{b}_j^{f+1}$. This equation holds true, since

$$\bar{b}_j^f \underset{\text{(line 40)}}{=} \bar{b}_k \underset{\text{(line 41)}}{=} b \underset{\text{(line 39)}}{=} \underline{b}_j^{f+1}.$$

Based on the sorting of arrival times and SoCs, see line 30, it follows that $\underline{t}_k \leq \underline{t}_{k'}$. We distinguish the two following cases: (1) $\underline{t}_k < \underline{t}_{k'}$ and (2) $\underline{t}_k = \underline{t}_{k'}$. In the first case, Claim 6 is applicable and it follows $\underline{b}_k \geq \underline{b}_{k'}$. Therefore, condition (ii) is fulfilled for this case, since

$$\bar{t}_j^f \underset{\text{(line 40)}}{=} \bar{t}_k \overset{\text{(Claim 4)}}{=} \underline{t}_k + \frac{1}{\alpha}(\bar{b}_k - \underline{b}_k) \underset{\text{(line 41)}}{=} \underline{t}_k + \frac{1}{\alpha}(b - \underline{b}_k) < \underline{t}_{k'} + \frac{1}{\alpha}(b - \underline{b}_{k'}) \underset{\text{(line 39)}}{=} \underline{t}_j^{f+1}.$$

If $\underline{t}_k = \underline{t}_{k'}$ it follows $\underline{b}_k \leq \underline{b}_{k'}$ based on the sorting in line 30. Assume $\underline{b}_k = \underline{b}_{k'}$ holds. It follows that $\bar{b}_k > \bar{b}_{k'}$, see line 30. However, this is a contradiction to $\bar{b}_k = b < \bar{b}_{k'}$, see line 38. Thus, $\underline{b}_k < \underline{b}_{k'}$ holds. The linear interpolation between lower and upper time-battery pair has a slope of α, see Claim 4. Thus, $\bar{t}_j^f < \underline{t}_j^{f+1}$ holds and condition (ii) is fulfilled.

(iii) Let $k \in D$ denote the first depot after the sorting in line 30. It follows,

$$l_j \overset{\text{(Claim 5)}}{\leq} \underline{t}_k \underset{\text{(line 35)}}{=} \underline{t}_j^1.$$

Let $\tilde{k} \in D$ denote the depot from which j is visited on the last curve element (denoted by q). It follows,

$$\bar{t}_j^q \underset{\text{(line 40)}}{=} \bar{t}_{\tilde{k}} \overset{\text{(Claim 5)}}{\leq} u_j.$$

(iv) We choose depot k and $\tilde{k}$ as in case (iii). It follows,

$$\underline{b}_j^1 \underset{\text{(line 35)}}{=} \underline{b}_k \overset{\text{(Claim 5)}}{\geq} c_j$$

and

$$\bar{b}_j^q \underset{\text{(line 40)}}{=} \bar{b}_{\tilde{k}} \overset{\text{(Claim 5)}}{\leq} B - c_j.$$

Thus, a time-battery curve is calculated at customer j, which concludes the inductive proof. □

Summing up, we have now shown that Algorithm 1 calculates a time-battery curve at each customer for a given feasible sequence.

Appendix B: Proof of Theorem 1

The proof of Theorem 1 is divided into two parts. Firstly, we show that if Algorithm 1 returns *feasible*, a feasible solution can be constructed and secondly, we prove that if Algorithm 1 returns *infeasible*, no feasible solution exists. Equally to Appendix A, we assume that all service times are zero. The proof of this theorem is divided into several lemmas, which we prove in the following.

Lemma 2. *Let s denote a customer sequence. If Algorithm 1 returns* feasible *it is possible to create a feasible solution for the given customer sequence.*

Proof. First, we show that Algorithm 1 does not violate any time window or battery bounds and then, present an extension of the algorithm in order to extract a feasible solution as stated in Definition 1.

A solution is considered infeasible if a customer's time window is violated or if the SoC is not within the bounds $[0, B]$. Algorithm 1 initializes the arrival time and SoC at the first customer with the earliest possible time and the corresponding SoC, see line 1. If the customer's time window is violated or the SoC is not sufficient, the algorithm returns *infeasible*, see line 2.

Based on Lemma 1, it holds that if Algorithm 1 returns *feasible* for sequence s, a time-battery curve is calculated at each customer of s. This means that based on the definition of a time-battery curve, the vehicle visits a customer only within the time window and the SoC upon arrival is within the bounds. It remains to show that the calculation of depot paths is also feasible, which means that the SoC remains within the bounds.

Consider an extension step from customer i to customer j by Algorithm 2. Based on the definition of a time-battery curve, it holds that the smallest SoC on the curve is sufficient to reach the depot closest to customer i, see Definition 2. Thus, at least one depot is evaluated by Algorithm 2. In line 4, it is checked that a depot is only evaluated if the time-battery curve at i contains an SoC value that is sufficient cover the required distance. It remains to show that the arrival time and SoC at the depot are correct. Let k denote an evaluated depot. The algorithm selects the smallest lower time-battery pair $(\underline{t}_i^f, \underline{b}_i^f)$ on the curve such that the corresponding upper time-battery pair is sufficient to reach k, see line 5. The departure time from i to k can be calculated as $\underline{t}_i^f + r + \tau_i$, where τ_i is the service time at i and r is given as $\frac{1}{\alpha}\max\{0, c_{i,k} - \underline{b}_i^f\}$. This calculation is valid, since $\bar{b}_i^f \geq c_{i,k}$ (line 5), the linear interpolation between lower and upper time-battery pair has a slope of α (Claim 4), and all time-battery pairs on the interpolation are feasible (Claim 5). The travel time from i to k is $c_{i,k}$. The sum of departure time and travel time represents the arrival time at k as defined in line 7. The SoC upon arrival at k is determined

by the SoC at j minus the travel time. The SoC at j can be expressed as $\underline{b}_j^f + \alpha r$, whereas r is defined as previously. Equally to the departure time at j, the selection of an SoC can be justified. Note that based on the definition of r, the SoC remains within the bounds. Thus, the SoC calculation in line 8 is correct.

In the depot to depot extension, see lines 10 to 21, a depot is only evaluated if the distance between the depots does not exceeds the battery capacity, see line 14. Remark, that in the problem description it is assumed that the reduced sub-graph of depots is connected with respect to B. The calculation of arrival times and SoCs is equally to the extension step from a customer to a depot with the difference that no service time is required at the depots and the interpretation of r changes. Now, it is the recharging time at a depot, which ensures that the vehicle's SoC remains within the bounds.

Before calculating arrival times at customer j, the algorithm excludes all depots that are either too far away from j or those for which the arrival time is already greater than the upper time window bound of j, see line 22. If all depots are excluded, the algorithm returns *infeasible*. Otherwise, the algorithm calculates for the remaining depots a lower and upper time-battery pair at j, see lines 25 to 29, which are used to construct the time-battery curve at j. The correctness of these calculations is proven in Claim 5. Thus, the calculations of Algorithms 1 and 2 are time and battery feasible. It remains to show how to construct a feasible solution from the algorithm's output.

In order to extract a feasible solution, see Definition 1, from Algorithm 1, some additional information have to be stored. At each evaluated depot, the predecessor depot or customer is stored. If the predecessor is a depot, the recharging time needed to reach the evaluated depot is captured, or if it is a customer, the curve element is memorized. In addition to that, the corresponding predecessor depot and recharging times are memorized for each element on the new time-battery curve.

Based on these additional information, a solution can be calculated by considering sequence s from the back to the front. The depot before the first customer of s and the depot after the last customer of s are trivially given by the respective closest depot and the recharging times are zero. To reconstruct a solution, we initially choose a curve element and its corresponding earliest pair of the time-battery curve at the last customer. Then, we create a depot path by considering the stored previous depots and corresponding recharging times until the previous customer is reached. This procedure is repeated until the first customer of s is reached.

□

After showing how to extract a solution from Algorithm 1, it remains to prove that no feasible solution exists if *infeasible* is returned, which is proven by mathematical induction. In order to state the proposition of the proof, we give a definition of *good* nodes. Definition 4 defines *good* depot nodes, and Definition 5 defines *good* customer nodes. In the following, let π' denote a solution, as defined in Definition 1, calculated by Algorithm 1 to a sequence s, and let π denote some given solution.

Definition 4. *Let s be a sequence, let $i, j \in s$ denote two sequential customers of s, and let π be a feasible solution to s. Further, let (t_k, b_k) be a time-battery pair of π at k, which is visited between i and j. We say that depot k is a* good *node if a time-battery pair (t'_k, b'_k), calculated by Algorithm 2, at the same depot following customer i exists, such that the following two inequalities hold:*

$$t_k \geq t'_k \tag{10}$$

$$b_k \leq b'_k + \alpha(t_k - t'_k) \tag{11}$$

This means that solution π' arrives earlier or equal at depot k compared to solution π. Moreover, if the arrival time difference is used to recharge the battery, the SoC in solution π' is greater or equal to the SoC of π. A similar definition is given for a customer. However, a direct comparison between the SoCs is used since no recharging is possible at customer nodes.

Definition 5. *Let s be a sequence, let i be a customer of s, and let π be a feasible solution to s. Further, let (t_i, b_i) be a time-battery pair of π at i. We say that customer i is a* good *node if a time-battery pair (t'_i, b'_i), calculated by Algorithm 1 or 2, on the time-battery curve at i exists, such that the following two inequalities hold:*

$$t_i \geq t'_i \tag{12}$$
$$b_i \leq b'_i \tag{13}$$

This means that solution π' arrives not later at customer i and with an SoC greater or equal compared to the pair of π. The proposition of the inductive proof is given by Proposition 1 and the inductive step is shown in Lemma 3 and 4.

Proposition 1. *Let s be a sequence, let π be a solution to s. If Algorithm 1 calculates a solution to s, then all depot and customer nodes contained in solution π are* good*, as defined in Definition 4 and 5.*

In the following, let (t, b) denote a time-battery pair of a feasible solution π, and let (t', b') denote a time-battery pair calculated by Algorithm 1 or Algorithm 2, respectively. Before we consider the correctness proof, we present auxiliary Claims 7 to 12, which define a lower bound for the arrival time at a customer or depot, and lower and upper bounds for the SoC upon arrival at a customer.

Claim 7. *Let i be a customer that is visited at time t_i with an SoC of b_i units, and let k be a depot that is visited immediately after i, whereas $c_{i,k}$ is less than b_i. It holds that the arrival time at depot k is lower bounded by $t_k \geq t_i + c_{i,k}$.*

Proof. The vehicle arrives at customer i at time t_i. The earliest time to depart from i is t_i since the service time is zero. The travel time from i to k is $c_{i,k}$ since the triangle inequality holds, it is also the shortest connection.

□

Claim 8. *Let k be a depot that is visited at time t_i, and let h be a depot or customer that is visited immediately after k. It holds that the arrival time at h is lower bounded by $t_h \geq t_k + r_k + c_{k,h}$.*

Proof. The vehicle arrives at depot k at time t_k. The earliest time to depart from k is $t_k + r_k$, where r_k correspond to the recharging time at k. The travel time from k to h is $c_{k,h}$, since the triangle inequality holds, it is also the shortest connection.

□

Claim 9. *Let k be a depot that is visited at time t_k with an SoC of b_k units, and let j be a customer that is visited immediately after k. It holds that arrival time at i is lower bounded by*

$$t_i \geq \max\{l_i, t_k + \frac{1}{\alpha}\max\{0, c_{i,k} + c_i - b_k\} + c_{i,k}\}.$$

Proof. The first part of the maximum is trivial since the vehicle is not allowed to arrive outside the time window at customer i. Based on Claim 8, the arrival time at i is lower bounded by $t_i \geq t_k + r_k + c_{i,k}$, where r_k correspond to the recharging time at the depot, and $c_{i,k}$ denotes the

travel time from depot k to i. The minimal recharging time is determined by the SoC needed to reach customer i plus the depot closest to i. This time is calculated as $\frac{1}{\alpha}\max\{0, c_{i,k} + c_i - b_k\}$. Thus, the lower bound of the claim is correct. Note that this term is used to calculate the earliest arrival time at a customer from each depot in Algorithm 2, see line 26.

□

Claim 10. *Let k be a depot that is visited at time t_k with an SoC of b_k units, and let i be a customer that is visited immediately after k. It holds that arrival time at i is lower bounded by*

$$t_i \geq \max\{l_i, t_k + \frac{1}{\alpha}(c_{i,k} + b_i - b_k) + c_{i,k}\}.$$

Proof. The first part of the maximum is trivial since the vehicle is not allowed to arrive outside the time window at customer i. The recharging time can be expressed as $r_k = \frac{1}{\alpha}(b_i + c_{i,k} - b_k)$, and together with Claim 8 the correctness follows.

□

Claim 11. *Let k be a depot that is visited at time t_k with an SoC of b_k units, and let h be a depot or customer that is visited immediately after k. It holds that the SoC upon arrival at h is given as $b_h = b_k + \alpha r_k - c_{k,h}$, where r_k denotes the recharging time at k.*

Proof. The SoC increases by α units for each time unit the vehicle recharges, up to a level of B units. Therefore, the SoC of the vehicle at the departure from k is given as $b_k + \alpha r_k$. The energy needed to travel from k to h is $c_{k,h}$. Thus, the equation of the claim holds.

□

Claim 12. *Let k be a depot that is visited at time t_k with an SoC of b_k units, and let i be a customer that is visited immediately after k. It holds that the SoC upon arrival at i is upper bounded by*

$$b_i \leq \min\{B - c_{i,k}, b_k + \alpha(t_i - t_k - c_{i,k}) - c_{i,k}\},$$

where t_i denotes the arrival time at i, and $c_{i,k}$ the travel time from depot k.

Proof. The first part of the minimum is trivial since the highest SoC at i is obtained after a full recharge at k and is given as $B - c_{i,k}$. For partial recharges at k, the recharging time can be expressed as $r_k = t_i - t_k - c_{i,k}$, and together with Claim 11 the correctness follows.

□

The following two lemmas provide the inductive step. First, we prove that if a customer node is *good*, then any depot, which precedes i in a solution π, is also *good*.

Lemma 3. *Let s denote a customer sequence, let i and $j \in s$ be two sequential customers, and let π be a solution to s. Suppose Algorithm 1 or 2 calculates a time-battery curve at i, then it holds that if customer i is good any depot $k \in D$ contained in solution π, which is visited between customers i and j, is also good.*

Proof. We prove Lemma 3 by mathematical induction.

Base case: Let k be the first depot following customer i in solution π. Let (t_i, b_i) denote the time-battery pair at which customer i is left for depot k, and let (t_k, b_k) denote the time-battery pair at depot k in solution π. If customer i is *good*, a time-battery pair (t'_i, b'_i) on the time-battery curve at customer i exists that fulfills inequalities 12 and 13. This means that $t'_i \leq t_i$ and $b'_i \geq b_i$ holds. Let (t''_i, b''_i) denote the departure time-battery pair at i to depot k calculated by Algorithm 2. It holds that $t''_i \leq t'_i \leq t_i$, since Algorithm 2 selects the earliest time, see line 5. Moreover, based on Definition 2 of a time-battery curve, it holds that $b'_i \leq b''_i + \alpha(t'_i - t''_i)$.

Let (t''_k, b''_k) denote the arrival time and SoC at depot k calculated by Algorithm 2. It follows that

$$t_k \overset{\text{(Claim 7)}}{\geq} t_i + c_{i,k} \overset{\text{(Prop.1)}}{\geq} t'_i + c_{i,k} \geq t''_i + c_{i,k} \underset{\text{(line 7)}}{=} t''_k \text{ and}$$

$$b_k = b_i - c_{i,k} \overset{\text{(Prop.1)}}{\leq} b'_i - c_{i,k} \overset{\text{(Def.2)}}{\leq} b''_i + \alpha(t''_i - t'_i) - c_{i,k} \overset{\text{(Claim 7)}}{\leq} b''_k + \alpha(t''_k - t_k).$$

Thus, depot k is *good*.

Inductive Step: Given two sequential depots $k-1, k$ following customer i in solution π. If depot $k-1$ is *good*, it holds that depot k is also *good*. Let (t_{k-1}, b_{k-1}) denote the time-battery pair at depot $k-1$ of solution π, and let (t'_{k-1}, b'_{k-1}) denote the time-battery pair at $k-1$ calculated by Algorithm 2. Proof by contradiction:

$$\begin{aligned}
&\text{Assume } \; t_k < t'_k \\
&\overset{\substack{\text{(Claim 8)} \\ \text{(lines 15 to 17)}}}{\Rightarrow} \; t_{k-1} + r_{k-1} + c_{k-1,k} < t'_{k-1} + \frac{1}{\alpha}\max\{0, c_{k-1,k} - b'_{k-1}\} + c_{k-1,k} \\
&\Rightarrow \; t_{k-1} + r_{k-1} < t'_{k-1} + \frac{1}{\alpha}\max\{0, c_{k-1,k} - b'_{k-1}\} \\
&\text{case (i) } c_{k-1,k} - b'_{k-1} \leq 0 \\
&\Rightarrow \; t_{k-1} + r_{k-1} < t'_{k-1} \qquad \lightning \\
&\text{case (ii) } c_{k-1,k} - b'_{k-1} > 0 \\
&\Rightarrow \; t_{k-1} + r_{k-1} < t'_{k-1} + \frac{1}{\alpha}(c_{k-1,k} - b'_{k-1}) \\
&\overset{\text{(Prop.1)}}{\Rightarrow} \; t_{k-1} + r_{k-1} < t'_{k-1} + \frac{1}{\alpha}(c_{k-1,k} - b_{k-1} + \alpha(t_{k-1} - t'_{k-1})) \\
&\Rightarrow \; t_{k-1} + r_{k-1} < t'_{k-1} + \frac{1}{\alpha}(c_{k-1,k} - b_{k-1}) + t_{k-1} - t'_{k-1} \\
&\Rightarrow \; r_{k-1} < \frac{1}{\alpha}(c_{k-1,k} - b_{k-1}) \\
&\Rightarrow \; \frac{1}{\alpha}(c_{k-1,k} - b_{k-1}) \leq \frac{1}{\alpha}\max\{0, c_{k-1,k} - b_{k-1}\} \leq r_{k-1} < \frac{1}{\alpha}(c_{k-1,k} - b_{k-1}) \qquad \lightning
\end{aligned}$$

This proves that Inequality 10 of Definition 4 remains satisfied. It remains to show that Inequality 11 is satisfied at k. Proof by contradiction:

$$\begin{aligned}
&\text{Assume } \; b_k > b'_k + \alpha(t_k - t'_k) \\
&\overset{\substack{\text{(Claim 11)} \\ \text{(line 18)}}}{\Rightarrow} \; b_{k-1} + \alpha r_{k-1} - c_{k-1,k} > b'_{k-1} + \alpha r'_{k-1} - c_{k-1,k} + \alpha(t_k - t'_k) \\
&\overset{\text{(Claim 8)}}{\Rightarrow} \; b_{k-1} + \alpha r_{k-1} > b'_{k-1} + \alpha r'_{k-1} + \alpha(t_{k-1} + r_{k-1} - t'_{k-1} - r'_{k-1}) \\
&\Rightarrow \; b_{k-1} > b'_{k-1} + \alpha(t_{k-1} - t'_{k-1}) \qquad \lightning
\end{aligned}$$

Thus, Proposition 1 holds at depot k, which concludes the induction. This inductive proof shows that Proposition 1 remains satisfied for each depot following customer i in a solution π.

□

In order to complete the inductive step, we still have to ensure that Proposition 1 remains satisfied in the extension from customer i to j.

Lemma 4. *Let s be a sequence, let i and j be two sequential customers of s, and let π be a solution to s. Algorithm 1 calculates a time-battery curve at i. If Proposition 1 is satisfied at customer $i \in s$, it remains satisfied at customer j.*

Proof. Based on Lemma 3, Proposition 1 holds at each depot following customer i contained in solution π. Assume depot $k \in D$ is the last depot visited in the solution π before visiting customer j. Let (t_k, b_k) and (t_j, b_j) denote the time-battery pair at k and j, respectively, in solution π. We show that there exists an time-battery pair (t'_j, b'_j) calculated by Algorithm 2 such that customer j is *good*. Let $\underline{b}'_j$ denote the lower SoC of at j coming from depot k calculated by Algorithm 2, see line 28, and let $\bar{b}'_j$ denote the upper SoC, respectively. First, we prove that value b_j is smaller than $\bar{b}'_j$.

$$
\begin{aligned}
b_j &\overset{\text{(Claim 12)}}{\leq} \min\{B - c_{j,k}, b_k + \alpha(t_j - t_k - c_{j,k}) - c_{j,k}\} \\
&\overset{\text{(Prop.1)}}{\leq} \min\{B - c_{j,k}, b'_k + \alpha(t_k - t'_k) + \alpha(t_j - t_k - c_{j,k}) - c_{j,k}\} \\
&\leq \min\{B - c_{j,k}, b'_k + \alpha(t_j - t'_k - c_{j,k}) - c_{j,k}\} \\
&\leq \min\{B - c_{j,k}, b'_k + \alpha(u_j - t'_k - c_{j,k}) - c_{j,k}\} \\
&\overset{\text{(line 29)}}{=} \bar{b}'_j
\end{aligned}
$$

We distinguish the two following cases: (i) $b_j \in [\underline{b}'_j, \bar{b}'_j]$ or (ii) $b_j < \underline{b}'_j$, and show that in both cases, there is a time-battery pair (t'_j, b'_j) calculated by Algorithm 2 that satisfies Proposition 1.

In case (i), we choose $b'_j = b_j$

$$
\begin{aligned}
\Rightarrow t_j &\overset{\text{(Claim 10)}}{\geq} \max\{l_j, t_k + \frac{1}{\alpha}(c_{j,k} + b_j - b_k) + c_{j,k}\} \\
&\overset{\text{(Prop.1)}}{\geq} \max\{l_j, t_k + \frac{1}{\alpha}(c_{j,k} + b'_j - b'_k - \alpha(t_k - t'_k)) + c_{i,k}\} \\
&= \max\{l_j, t'_k + \frac{1}{\alpha}(c_{j,k} + b'_j - b'_k) + c_{j,k}\} \\
&\overset{(\diamondsuit)}{=} t'_j
\end{aligned}
$$

Equation $(\diamondsuit)$ holds since $b_j \in [\underline{b}'_j, \bar{b}'_j]$, and Claim 5 states that all time-battery pairs on the linear interpolation between the lower and upper pair are obtainable. Moreover, Algorithm 2 avoid unnecessary idle times at a depot.

In the second case (ii), we choose $b'_j = \underline{b}'_j$. If $t_j = l_j$ holds, we choose $t'_j = l_j$ and Proposition 1 holds. This assumption is valid since Algorithm 2 considers the earliest possible arrival time, and Proposition 1 holds at depot k.

This leaves to show the case of $t_j > l_j$. Let $\underline{t}'_j$ denote the lower time at customer j from

depot k. It follows:

$$
\begin{aligned}
t'_j = \underline{t}'_j &\underset{\text{(line 26)}}{=} \max\{l_j, t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b'_k\} + c_{j,k}\} \\
&\text{case (1) } l_j > t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b'_k\} + c_{j,k}: \\
&\Rightarrow t'_j = l_j < t_j \\
&\text{case (2) } l_j \leq t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b'_k\} + c_{j,k}: \\
&\Rightarrow t'_j = t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b'_k\} + c_{j,k} \\
&\overset{(\diamondsuit\diamondsuit)}{=} t'_k + c_{j,k} \\
&\overset{\text{(Prop.1)}}{\leq} t_k + c_{j,k} \\
&\leq t_j
\end{aligned}
$$

Equation $(\diamondsuit\diamondsuit)$ holds because $\underline{t}'_j$ is greater than l_j, b_j is feasible, and $\underline{b}'_j$ is the smallest battery calculated by Algorithm 2 to reach customer j. Thus, r'_k is zero, otherwise the b'_j can be decreased by some ϵ, with $\epsilon > 0$.

Note that the considered time-battery pair (t'_j, b'_j) is not necessarily on the time-battery curve because it can be dominated by a time-battery pair (t''_j, b''_j) on the curve, with $t''_j \leq t'_j$ and $b''_j \geq b'_j$. In this case, Proposition 1 also holds for time-battery pair (t''_j, b''_j).

□

This proves that Proposition 1 remains satisfied from customer i to j if a time-battery curve is given at customer i. In the following, we combine the statements of Lemmas 1, 3, and 4.

Lemma 5. *Let s be a customer sequence. Proposition 1 is satisfied at each customer of s.*

Proof. This lemma is proven by mathematical induction. Lemma 1 satisfies that Algorithm 1 calculates a time-battery curve at each customer of s if the decision problem is stated *feasible*. The base case is to show that at the first customer, a time-battery curve is constructed, and Proposition 1 holds at the first customer of sequence s. This is trivially fulfilled since Algorithm 1 initializes the first customer from his nearest depot as early as possible, see line 1, and both solutions start with a fully charged battery. The inductive step is given by Lemmas 3 and 4.

□

It still remains to show that if Algorithm 1 returns *infeasible*, no feasible solution exists.

Lemma 6. *Let s be a customer sequence. If Algorithm 1 does not find a feasible solution to s, then no feasible solution exists.*

Proof. Suppose a solution π to sequence s exists, and Algorithm 1 returns *infeasible*. The stop criterion is either called in Algorithm 1 or in its subroutine (Algorithm 2). Algorithm 1 returns *infeasible* if the first customer in s is not reachable, see line 2. However, Algorithm 1 starts as early as possible, with an SoC of B, from the nearest depot, see line 1. Thus, no other solution can be feasible.

In the following, we assume that the stopping criterion is called in Algorithm 2 in the extension from i to j, with $i, j \in s$. Note that at all depots of the depot path between i and j in π, Proposition 1 holds. This implies that Algorithm 2 calculates a time-battery pair at all these depots. The stopping criterion called in the two following cases:

(1) the reduced set of depots D' to evaluate customer j is empty (line 23) or

(2) the earliest arrival time at customer j is not within the time window of j (line 32).

Case (1): There are two possible reasons for a depot being excluded from D'. Either the arrival time at the depot is already later than the upper time window bound of customer j, or the distance to reach j plus the distance of the depot closest to j exceeds the battery capacity B. Let k denote the last depot before customer j in solution π, and suppose that k is excluded from set D'. At depot k Proposition 1 holds. Thus, depot k cannot be part of a feasible solution.

Case (2): Since Proposition 1 holds, it follows that depot k is a *good* node, which means that Inequality 10 and 11 are satisfied. Let $\underline{t'_j}$ denote the earliest arrival time from depot k calculated by Algorithm 2. It follows:

$$
\begin{aligned}
t_j &\leq u_j \\
&< \underline{t'_j} \\
&\underset{\text{(line 26)}}{=} \max\{l_j, t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b'_k\} + c_{j,k}\} \\
&= \max\{l_j, t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b'_k\} + c_{j,k}\} \\
&\overset{\text{(Prop.1)}}{\leq} t'_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b_k + \alpha(t_k - t'_k)\} + c_{j,k} \\
&\leq \max\{l_j, t_k + \frac{1}{\alpha}\max\{0, c_{j,k} + c_j - b_k\} + c_{j,k}\} \\
&\overset{\text{(Claim 9)}}{\leq} t_j \qquad \lightning
\end{aligned}
$$

It follows that if Algorithm 1 returns *infeasible*, then no solution exists.

□

It remains to show that the runtime of Algorithm 1 is polynomial bounded.

Claim 13. *Let s be a customer sequence with S customers and let d denote the total number of depots. Algorithm 1 has an asymptotic runtime of $O(S \cdot d^2)$, which is polynomial in the input size.*

Proof. Algorithm 2 is called for each pair of consecutive customers (S-1 times) and has a runtime of $O(d^2)$. Therefore, the total runtime of Algorithm 1 is bounded by $O(S \cdot d^2)$.

□

Now, we are ready to prove Theorem 1.

Proof. Proof of Theorem 1: Lemma 2 states that if Algorithm 1 returns feasible a feasible solution can be constructed. Lemma 6 states that if Algorithm 1 returns *infeasible*, then no feasible solution exists. Both lemmas together show the correctness of the algorithm. The polynomial runtime is proven by Claim 13.

□

Appendix C: Dynamic Program for Backward Extensions

Algorithm 3: Backward extension from customer i to customer j

Input: customers i, j, depot set D, time-battery curve $(\underline{t}_i^f, \underline{b}_i^f), (\bar{t}_i^f, \bar{b}_i^f), f = 1, \ldots, q$

$F \leftarrow \emptyset$
$t_k \leftarrow 0, b_k \leftarrow \infty, \forall\ k \in D$
\# evaluate depots from customer i
for $k \in D$ *with* $c_{i,k} \leq B - \underline{b}_i^1$ **do**
  $(t, b) \leftarrow \max\{(\bar{t}_i^f, \bar{b}_i^f) : B - \underline{b}_i^f \geq c_{i,k}, f \in \{1, \ldots, q\}\}$
  $r \leftarrow \frac{1}{\alpha} \min\{0, c_{i,k} - (B - b)\}$
  $t_k \leftarrow t - r - c_{i,k}$
  $b_k \leftarrow b - \alpha r + c_{i,k}$
end
\# depot to depot evaluation
while $F \neq D$ **do**
  $k \leftarrow \arg\max\{t_w : w \in D \setminus F\}$
  $F \leftarrow F \cup \{k\}$
  for $h \in D \setminus F$ *with* $c_{k,h} \leq B$ **do**
    $r \leftarrow \frac{1}{\alpha} \max\{0, c_{k,h} - (B - b_k)\}$
    if $t_k - r - c_{k,h} > t_h$ **then**
      $t_h \leftarrow t_k - r - c_{k,h}$
      $b_h \leftarrow b_k - \alpha r + c_{k,h}$
    end
  end
end
$D' \leftarrow \{k \in D : t_k \geq l_j, c_{j,k} + c_j \leq B\}$
if D' *is empty* **then**
  Return: Extension not possible
end
\# calculate lower and upper times-battery pairs at customer j
$\bar{t}_k \leftarrow \min\{u_j, t_k - \frac{1}{\alpha} \max\{0, c_{j,k} + c_j - (B - b_k)\} - c_{j,k} - \tau_j\}$, for $k \in D'$
$\underline{t}_k \leftarrow \min\{\bar{t}_k, \max\{l_j, t_k - \frac{1}{\alpha} b_k - c_{j,k} - \tau_j\}\}$, for $k \in D'$
$\bar{b}_k \leftarrow \max\{c_{j,k}, b_k - \alpha(t_k - c_{j,k} - \tau_j - \bar{t}_k) + c_{j,k}\}$, for $k \in D'$
$\underline{b}_k \leftarrow \max\{c_{j,k}, b_k - \alpha(t_k - c_{j,k} - \tau_j - l_j) + c_{j,k}\}$, for $k \in D'$
sort depots in decreasing order by $\bar{t}_k$, if tie increasing by $\bar{b}_k$, if still tie increasing by $\underline{b}_k$
rename depots according to new order from 1 to d'
if $\bar{t}_1 < l_j$ **then**
  Return: Extension not possible
end
\# built the time-battery curve at customer j
$(\bar{t}_j^1, \bar{b}_j^1) \leftarrow (\bar{t}_1, \bar{b}_1)$
$(\underline{t}_j^1, \underline{b}_j^1) \leftarrow (\underline{t}_1, \underline{b}_1)$
$b \leftarrow \underline{b}_1$
$f \leftarrow 2$
for $k = 2$ *to* d' **do**
  if $\bar{t}_k \geq l_j$ *and* $\underline{b}_k < b$ **then**
    $(\bar{t}_j^f, \bar{b}_j^f) \leftarrow (\bar{t}_k - \frac{1}{\alpha}(\bar{b}_k - b), b)$
    $(\underline{t}_j^f, \underline{b}_j^f) \leftarrow (\underline{t}_k, \underline{b}_k)$
    $b \leftarrow \underline{b}_k$
    $f \leftarrow f + 1$
  end
end
Return: time-battery curve of customer j

References

[1] Niels Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981, 2018. doi:10.1287/trsc.2017.0791.

[2] Pitney Bowes. Pitney bowes parcel shipping index reports continued growth as global parcel volume exceeds 100 billion for first time ever. Accessed April 17, 2022, 2020. URL http://news.pb.com/article_display.cfm?article_id=5958.

[3] Nils Boysen, Stefan Schwerdfeger, and Felix Weidinger. Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085–1099, 2018. doi:10.1016/j.ejor.2018.05.058.

[4] Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43(1):1–58, 2020. doi:10.1007/s00291-020-00607-8.

[5] J-F Cordeau, G Laporte, and A Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001. doi:10.1057/palgrave.jors.2601163.

[6] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. *Column Generation*. Springer, Boston, MA, 2005. doi:10.1007/b135457.

[7] Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016. doi:10.1287/opre.2016.1535.

[8] Jacques Desrosiers and Marco E. Lübbecke. *A Primer in Column Generation*, pages 1–32. Springer US, Boston, MA, 2005. doi:10.1007/0-387-25486-2_1.

[9] Deutsche Post DHL Group. Green logistics. Accessed April 17, 2022, 2017. URL https://www.dhl.com/global-en/delivered/sustainability/zero-emissions-by-2050.html.

[10] Tomislav Erdelić and Tonči Carić. A survey on the electric vehicle routing problem: Variants and solution approaches. *Journal of Advanced Transportation*, 2019:1–48, 2019. doi:10.1155/2019/5075671.

[11] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, 2018. URL http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

[12] Stefan Irnich and Guy Desaulniers. *Shortest Path Problems with Resource Constraints*, pages 33–65. Springer US, Boston, MA, 2005. doi:10.1007/0-387-25486-2_2.

[13] Merve Keskin and Bülent Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127, 2016. doi:10.1016/j.trc.2016.01.013.

[14] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi:10.1287/opre.1050.0234.

[15] Huiting Mao, Jianmai Shi, Zhongbao Zhou, and Long Zheng. The unmanned aerial vehicle routing problem with recharging. *Unpublished report*, 2021. doi:10.12074/202101.00070.

[16] Francesco Marra, Guang Ya Yang, Chresten Træholt, Esben Larsen, Claus Nygaard Rasmussen, and Shi You. Demand profile study of battery electric vehicle under different charging options. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, 2012. doi:10.1109/PESGM.2012.6345063.

[17] Chase C. Murray and Amanda G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015. doi:10.1016/j.trc.2015.03.005.

[18] Junhyuk Park, Jongsung Lee, SeHwan Ahn, Jungryul Bae, and Hyunchul Tae. Exact algorithm for the capacitated team orienteering problem with time windows. *Mathematical Problems in Engineering*, 2017:1024–1230, 2017. doi:10.1155/2017/9285403.

[19] Moritz Poeting, Stefan Schaudt, and Uwe Clausen. A comprehensive case study in last-mile delivery concepts for parcel robots. In Navonil Mustafee, Ki-Hwan G. Bae, Sanja Lazarova-Molnar, Markus Rabe, and Claudia Szabo, editors, *Proceedings of the Winter Simulation Conference*, page 1779–1788. IEEE Press, 2019. doi:10.1109/WSC40007.2019.9004811.

[20] Sepideh Pourazarm, Christos G. Cassandras, and Tao Wang. Optimal routing and charging of energy-limited vehicles in traffic networks. *International Journal of Robust and Nonlinear Control*, 26(6):1325–1350, 2016. doi:10.1002/rnc.3409.

[21] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014. doi:10.1287/trsc.2013.0490.

[22] Marc-Oliver Sonneberg, Max Leyerer, Agathe Kleinschmidt, Florian Knigge, and Michael H. Breitner. Autonomous unmanned ground vehicles for urban logistics: Optimization of last mile delivery operations. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, pages 1538–1547, 2019. doi:10.24251/HICSS.2019.186.

[23] Timothy M. Sweda, Irina S. Dolinskaya, and Diego Klabjan. Optimal recharging policies for electric vehicles. *Transportation Science*, 51(2):457–479, 2017. doi:10.1287/trsc.2015.0638.

[24] Hyunchul Tae and Byung-In Kim. A branch-and-price approach for the team orienteering problem with time windows. *International Journal of Industrial and Systems Engineering*, 22(2):243 – 251, 2015.

[25] Chien-Ming Tseng, Chi-Kin Chau, Khaled M. Elbassioni, and Majid Khonji. Flight tour planning with recharging optimization for battery-operated autonomous drones. *CoRR*, 2017. URL http://arxiv.org/abs/1703.10049.

[26] Daniela Rojas Viloria, Elyn L. Solano-Charris, Andrés Muñoz-Villamizar, and Jairo R. Montoya-Torres. Unmanned aerial vehicles/drones in vehicle routing problems: a literature review. *International Transactions in Operational Research*, 28:1626–1657, 2020. doi:10.1111/itor.12783.