ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Analytics and Operational Research in Logistics

A p-step formulation for the Electric Vehicle Routing Problem

Author:
Nienke Veldkamp (475284)
Supervisor:
R. Spliet
Second assessor:
T. Dollevoet

Abstract

In this thesis, we introduce the *p*-step formulation of the Electric Vehicle Routing Problem (EVRP). This formulation is a generalisation of both the arc flow and the set partitioning formulation applied to a general Electric Vehicle Routing Problem with recharging stations. In this paper, we first give the complete formulation and show how to solve it with a restricted master problem with two different options for a pricing problem. Both pricing problems solve the elementary resource constraint shortest path problem, where the battery capacity is restricted and the length of the path is restricted. The first way we solve the pricing problem is by constructing the integer programming formulation and putting this into a solver. The second way we solve the pricing problem is by implementing a bidirectional labelling algorithm. Which we use to quickly identify negative reduced cost variables which can be added to the linear relaxation of the p-step formulation. Furthermore, our results show that there are computational improvements to be gained from using a column generation algorithm in combination with a *p*-step formulation, as opposed to Arc-flow and set-partitioning. Lastly, we show that our formulation can be easily extended on to cover more intricate types of vehicle routing problems.

Keywords: *Thesis, VRP, EVRP, Green Logistics, p-step*

May 11, 2023

Contents

1 Introduction

In our present-day society, climate change and the greenhouse effect have become a hot political topic. One of the recent reports of the European Commission showed that 27% of the CO_2 emissions are caused by road transport (EEA 2021). Furthermore, they also note that, over the last two decades, while emissions reasonably decreased in other sectors they strongly increased in the transport sector. To counter this, several governments across the world have implemented policies and legislation to minimize car emissions. These political decisions have had a significant impact on the logistics industry. This is why companies have incorporated *Green logistics* projects to reduce CO_2 emissions.

Reducing emissions can be achieved in two ways. The first way is through better exploitation of existing resources. By applying more efficient and sophisticated route planning optimization methods and adopting a smart distribution system it would be possible to achieve this. This would lead to a decrease in travelling distance and thus reduce emissions(F. Mancini 2013; S. Mancini 2013). The second way is by introducing innovative, environmentally friendly technologies. One of the most adapted options in logistics for this is the use of Electric Vehicles (EVs) in freight deliveries. The second way is, in general, preferred since the first way only leads to a few percent reduction in emission and the emission level of the vehicles remains high. Moreover, EVs have several advantages over conventional vehicles, aside from being cleaner: (i) they are more energy efficient: (ii) they can be powered from renewable energy sources; and (iii) they are independent of the fluctuating oil price (S. Mancini 2017).

For VRPs the exact approaches that are based on integer or mixed integer programming have the most success. In the literature there are two formulations that dominate this topic. The first is an arc-based formulation, that makes use of binary variables to indicate whether an arc is used or not. The second is a path-based formulation, which used binary variables to indicate whether a certain path (route) is used. Since both formulations have their strong and weak points, p-step programming is introduced. This provides a formulation family that includes both traditional arc-based and path-based formulations. As a result, it is a generalization that combines arc-based and path-based formulations while also introducing new formulations (Dollevoet et al. 2020).

For all the aforementioned reasons we decided to focus on creating a p-step formulation for the electric vehicle routing problem. The Capacitated Vehicle Routing Problem (CVRP) is a classic optimisation problem. Because of its practical relevance and inherent difficulty, the VRP is one of the most studied problems in the field of combinatorial optimization. The Electric Vehicle Routing Problem (EVRP) is in essence the same problem, the only difference being the type of vehicles. It is the problem of meeting customer demands from a depot using homogeneous electrical vehicles with limited capacity at the lowest possible cost while visiting each customer only once. In addition to its numerous extended versions, it has many practical applications and is extensively researched, see Toth and Vigo (2014).

This thesis is organised as follows. In Section 2 we will elaborate on the relevant literature for the Electric Vehicle Routing Problem. Then we give a formal problem description in Section 3. In Section 4, we introduce the methods used in this thesis. We provide the results of our computational experiments in Section 6. Finally, we conclude our thesis in Section 7.

2 Literature Review

As this thesis is mainly concerned with the problem of finding a formulation for the p-step Electric Vehicle Routing Problem, we will give a compact overview of relevant scientific literature relating to electric vehicle routing problems. First we need to note that there are two types of EVs. There is the battery electric vehicle, which only uses power from a battery in the vehicle. Then there is also a hybrid electric vehicle, which is partially powered by the battery inside the vehicle and partially by other sources. In this paper we will focus on the battery electric vehicle, which from this point on will be referred to as electric vehicles or EVs. The additional problems EVs bring to the VRP are, the limited driving range and the need for intermediate recharging. The range that EVs can deliver on a fully charged battery is about 160 to 240 kilometres (Feng and Figliozzi 2013). Whereas conventional vehicles have a range of approximately 480 to 650 kilometres (Young et al. 2013). Thus to accomplish a similar driving distance, EVs have to visit charging stations more frequently.

2.1 Electric Vehicle Routing

Although limited, the EVRP in the literature can be viewed best as a variant on the Green Vehicle Routing Problem (GVRP) proposed by Erdoğan and Miller-Hooks (2012). They look at a vehicle routing problem that considers alternative fuel vehicles, which also includes electric vehicles, with a limited travel distance that must be recharged during routing. Like the normal VRP, the GVRP visits each customer exactly once. However, instead of minimizing the costs the GVRP minimises the total travel distance of each vehicle. Also, the recharging stations can be visited as many times as necessary by any vehicle in the fleet. Which differs from the original VRP. To allow those multiple (and possible no) visits to the vertices for the recharging stations while requiring exactly one visit to the other vertices, the graph is augmented with a set of dummy vertices, one for each potential visit to a charging station or the depot serving as a charging station. As a result, the conventional VRP formulation can be used for EVRP with only minor modifications; the only change to the conventional VRP formulation is the addition of a constraint requiring that each charging/refuelling station be visited only once. Additionally, the parameter indicating the allowed distance is changed. In the VRP each node can only be visited once, while we want charging stations to be allowed to be visited multiple times.

A more recent paper by Lin et al. (2016) presents a general Electric Vehicle Routing Problem (EVRP) that determines the optimal routing strategy with the least amount of travel time, energy cost, and number of EVs dispatched. Furthermore, this is the first EVRP model to take into account the effect of vehicle load on battery consumption. Like the aforementioned GVRP, the EVRP considers unrestricted re-charging activities at charging stations. They find that compared to conventional vehicle VRPs, the EVRP has similar travel time and distance but long en-route re-charging time, which translates into a considerable amount of additional labour cost. Moreover, they find that the relative distribution of charging stations to customer points greatly affects the routing strategies

For a complete survey of recent developments in the EVRP, the readers may refer to Erdelić

and Carić (2019). They provide the challenges that have arisen as a result of the incorporation of electric vehicles into delivery processes are described, as well as electric vehicle characteristics and recent energy consumption models. Several EVRP variants and related problems are observed. Efficient VRP heuristics and metaheuristics had to be adapted to deal with the new routing challenges in EVRP. An overview of cutting-edge procedures for dealing with the EVRP and related issues is provided.

Since the survey covers a big variety of related problems we take a look at applied procedures of the papers with the most similar problems. The research of Pourazarm et al. (2014) compares a Mixed Integer Program (MIP) with a Dynamic Programming (DP) approach to solve the EVRP with the possibility of recharging, where they keep in mind both (i) homogeneous and (ii) in-homogeneous charging functions. For which in case (ii) they find that the DP, results in optimal solutions with lower computational complexity compared to the MIP for the smaller instances. For larger instances the DP is outperformed by the MIP.

The green vehicle routing problem discussed by Koç and Karaoglan (2016) uses a heuristic based exact approach. The Exact algorithm is based on the branch-and-cut algorithm, which combines several valid inequalities from the literature to improve lower bounds and introduces a heuristic algorithm based on simulated annealing to obtain upper bounds. The solution approach is measured in terms of the number of test instances solved to optimality, bound quality, and computation time required to arrive at the best solution to the various test problems. Their results show that 55% of the instances with 20 customers can be solved to optimality solved in a reasonable amount of time, which is about 20 to 30 minutes.

Wen et al. (2016) uses both an exact solver and a heuristic on small instances but only uses the heuristic for the larger instances. ALNS is able to find high quality solutions to the small instances containing up to 30 trips. The computational time of ALNS is just a fraction of the time needed CPLEX for the more challenging instances. The comparison to the MD-VSP results shows that solving the MD-VSP often does not provide a tight lower bound to the E-VSP. This is caused by the restricted driving range in the E-VSP that forces vehicles to recharge which increases deadheading and decreases utilization. The solutions are found within a few seconds for the small instance and a few minutes for the larger instances.

In Zhang, Gajpal, and Appadoo (2018) two solution methods are proposed; the two-phase heuristic algorithm and the meta-heuristic based on ant colony system (ACS). Both methods find solutions to the test instances within reasonable time, less than 30 minutes. They do find that the ACS algorithm is superior to the two-phase heuristic in obtaining the high-quality solutions, but it requires more computation time.

For the EVRP proposed in Zhang, Gajpal, Appadoo, and Abdulkader (2018) they also look at two solution methods and compare these with the outcome of a CPLEX solver. The first method is an Adaptive large neighbourhood search and the second is an Ant Colony (AC) algorithm similar to the ACS mentioned in the aforementioned paper. The objective in their EVRP is to minimize the energy consumption of electric vehicles. For the small instances they find that the objective and runtime of the AC algorithm outperforms the ALNS, with an average gap of 3.2% and runtime of 0.9 seconds compared to 10% and 10.2 seconds respectively. For the larger instances they also find that the AC algorithm outperforms the ALNS with finding

an answer in under ten minutes on average.

Lastly, we look at the results of the path based green vehicle routing problem proposed by Bruglieri et al. (2019). They seek to minimize total travel distance by routing Alternative Fuel Vehicles based at a depot. To solve their problem they provide a two-phase exact approach, with each route composed of two or more paths. Each route serves a subset of customers without the need for intermediate refuelling. They find that their approach outperforms existing exact methods on benchmark instances and can be generalized to solve other VRPs with Intermediate Stops. The average run time of their two-phase path method only takes about 6 minutes while the existing exact formulation and branch-and-cut method take more than 30 minutes to solve.

2.2 P-step Formulation

For VRPs the exact approaches that are based on integer or mixed integer programming have the most success. In the literature, two formulations dominate this topic. The first is an arc-based formulation, that makes use of binary variables to indicate whether an arc is used or not. The second is a path-based formulation, which used binary variables to indicate whether a certain path (route) is used. To get "the best of both worlds" the p-step formulation can be used.

	Arc flow	Set Partitioning
LP	weak	**strong**
CPU	**short**	long

Table 1: Comparison of the arc-flow and set-partitioning formulations

The P-step formulation for VRPs is a relatively new way of formulating the classic VRP. The p-step formulation is a generalisation of the acr-flow and set-covering formulation. In Dollevoet et al. (2020), the p-step formulation is introduced for the Capacitated Vehicle Routing Problem. They show that, although not monotonically, the LP-bound of the p-step formulation is increasing in p. By proving that the set partitioning formulation is NP-hard they show that there does not a strongest compact formulation for the CVRP if $P \neq NP$. They provide a new strongest formulation for the CVRP with a number of variables and constraints limited by a fixed degree polynomial. Which is the p-step formulation with a degree three and higher. Moreover, their results indicate that there are computational advantages from using the p-step formulation, as opposed to the traditional arc-based and path-based formulations.

In Rabbie (2018) they describe the application of an exact algorithm to the solution of a pricing problem arising from the use of column generation in the context of the capacitated vehicle routing problem. The p-step formulation is a generalization of the vehicle flow and set covering formulations. As a result, the corresponding pricing problem is modelled as an elementary resource-constrained shortest-path problem with a limited number of customers on a path. They solve their pricing problem by making use of a pulse algorithm. Which they, in particular, use for quickly identifying negative reduced cost variables which can be added to the linear relaxation of the p-step formulation.

2.3 Relation of our Research to the Literature

The core of the research done in this paper is based on Dollevoet et al. (2020), who introduced the p-step formulation for Capacitated Vehicle Routing Problems (CVRP). This p-step formulation can be seen as an arc-flow formulation as well as a set partitioning formulation or anything in between depending on the size of p.

3 Problem Description

Although the EVRP is a relatively well-known problem we give a short description in Section 3.1. This is for the sake of completeness as well as to introduce the notations we use in this thesis. Then in Section 3.2 we specify p-steps and provide a new integer programming formulation of the EVRP which makes use of these p-steps. In Section 3.3 we describe how we obtain a finite formulation. Furthermore, we provide the complete formulation we use to solve the EVRP in Section 3.4.

3.1 The electric vehicle routing problem

Consider an undirected graph $G = (V, E)$. Where $V = N \bigcup S$ with $N = \{0, ..., n+1\}$ is a set of locations with 0 representing the starting depot, $n+1$ representing the ending depot and $N' = \{1, ..., n\}$ the set of customers and $S = \{n+2, n+3, ..., n+s, \}$ representing the charging stations. $E = \{(i, j) : i, j \in N, i \neq j, i \neq n+1, j \neq 0\}$. An unlimited amount of vehicles with battery capacity Q is available. The rate at which the battery decreases is τ, which measures the consumption rate per km. Each vehicle traverses an elementary path from 0 to $n+1$, which we refer to as a route, to satisfy the demand of all customers along the path.

Given these assumptions we can depict an example solution for a set customers $N' = \{C1, ..., C10\}$ and charging stations $S = \{S1, .., S4\}$, this is shown in Figure 1. In this example solution we have three identical vehicles that are 100% charged when they set out to their route. The numbers on the arcs represent the battery percentage that is left after traversing an arc. Additionally we assume that the vehicles are fully charged at the stations thus after visiting the stations the battery will have 100%.

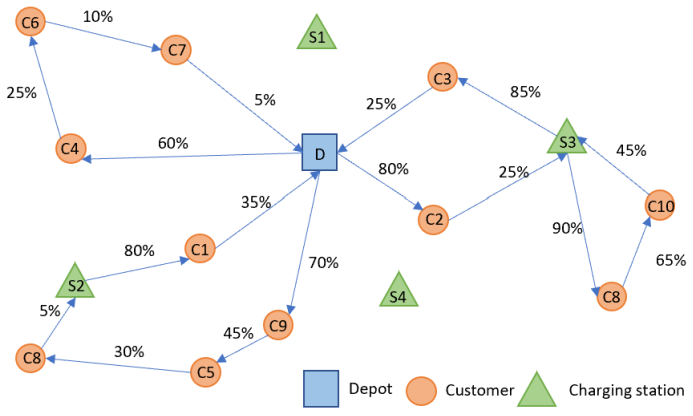

Figure 1: EVPR solution example

3.2 Integer programming formulation

Given the notation, we show a new integer programming formulation. Consider a p-step r to be a pair (P_r, δ_r). Where P_r represents an elementary path in G that i) either travels over exactly p arcs, ergo visiting $p+1$ nodes, or ii) starts at 0 and travels over at most p arcs. Additionally, δ_r represents the energy left over after visiting all customers on a route prior to arriving at the first location on P_r. Thus δ_r is referred to as the prior energy left over of r and we denote $q(r)$ as the total energy left over of visiting the customers on the p-step r. A p-step is feasible if $q(r) \geq 0$ and in the case, there is no recharge on r the condition that $\delta_r > q(r)$ also needs to be fulfilled.

In the formulation we use, we connect p-steps to represent routes. These steps can only be connected if their start/end locations and battery percentages matches. If we want to connect two p-steps r and s, we need location $i \in N'$ to be the last location in r and first of s, and the total battery percentage after leaving i on r to match that of s, that is $q(r) = q_i$. Where q_i is the battery percentage left after the first stop. Furthermore, in our formulation, we impose that each customer can only be visited once. By doing this, we guarantee that any connection resulting in a path from 0 to n+1 is elementary and satisfies the battery capacity constraint, hence represents a route.

Next, we need to make sure the p-steps are connected well. To represent the sequence of p-steps in our formulation, we introduce e_r^i and q_r^i. Let e_r^i take a value of 1, if i is the first location on P_r, a value of -1 if i is the last location on P_r and a value of 0 otherwise. Additionally, let q_r^i which is the leftover battery after trips until now be $-q_i$ if i is the first location on P_r, $q(r)$ if i is the last location on P_r and 0 otherwise.

Lastly, let x_r be a binary p-step variable, that indicates whether p-step r is used or not. They have a corresponding used battery percentage d_r, where d_r is given by $\sum_{(i,j)\in P_r} d_{ij}$ which is the used battery on each arc. The EVRP can then be formulated as follows.

$$\min \sum_{r \in R^p} d_r x_r \tag{1}$$

$$s.t. \sum_{r \in R^p} a_r^i x_r = 2 \qquad \forall i \in N' \tag{2}$$

$$\sum_{r \in R^p} e_r^i x_r = 0 \qquad \forall i \in N' \tag{3}$$

$$\sum_{r \in R^p} q_r^i x_r = 0 \qquad \forall i \in N' \tag{4}$$

$$x_r \in \{0, 1\} \qquad \forall r \in R \tag{5}$$

The total battery used is minimized in the objective function (1). Constraints (2) ensure that every customer is visited exactly once. The constraints (3) and (4), make sure the problem gets connected. They ensure that if a p-step is chosen and ends at node i then also a p-step that starts at node i is selected with adequate prior demand. Lastly, constraints (5) represent the binarity of variables x on the p-step. We will refer to this formulation ((1)-(5)) as the p-step formulation.

3.3 Finite formulation

Since the leftover battery percentage δ_r is continuous, the number of p-steps in R^p is unlimited and by extension so are the number of variables of x_r. This is the case unless $q_i = 100\%$ for all $i \in N'$. Nonetheless, it is possible to redefine the p-step formulation only to include a limited number of p-steps. We can achieve this by only considering p-steps for which the prior battery usage is minimal or maximal. By doing this all other p-steps can be represented as a convex combination of these two extremes. To obtain the minimal and maximal battery percentage needed we first determine the p-step. After finding the p-step, we know the distance between each node we visit and thus how much battery percentage we need to traverse each edge. We can calculate the minimal and maximal battery we need to start this p step with from that. Then, if we visit a charging station on the p-step, we only need to calculate usage until the charging station is visited. This is because after a charging station, the battery will always be 100%. To further clarify this concept we give an illustration in Figures 2 and 3.

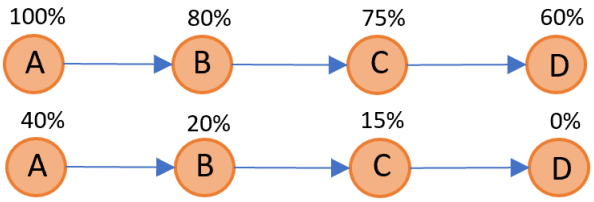

Figure 2: Minimal and maximal battery usage of a p-step without visiting a recharge station

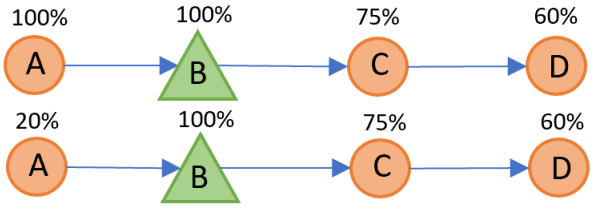

Figure 3: Minimal and maximal battery usage of a p-step with visiting a recharge station

In order to be more specific we first note that the p-step formulation will remain valid if the equalities (4) are replaced by

$$\sum_{r \in R^p} q_r^i x_r \geq 0 \qquad \forall i \in N' \tag{6}$$

When (6) is a strict inequality, it can be interpreted as having additional power which is mistakenly considered as less power by some p-step. The vehicle will return with the additional $\Delta\%$ at the depot. Besides, this does not affect the LP bounds, as we can lower the mistaken power to obtain a new solution for which (6) are satisfied with equality without changing the objective value.

Additionally, binary conditions (5) for the p-step variables are replaced by the arc-flow version of the binary conditions. For this, we introduce the binary arc-flow variable θ_{ij} which indicated whether arc $(i, j) \in A$ is used. Additionally, let b_{ij}^r be a binary parameter that

indicates whether arc $(i,j) \in A$ is used by p-step r. This results in (5) being replaced by

$$\sum_{r \in R^p} b_{ij}^r x_r = \theta_{ij} \qquad \forall (i,j) \in E \tag{7}$$

$$x_r \geq 0 \qquad \forall r \in R^p \tag{8}$$

$$\theta_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{9}$$

By using constraints (7) and (9) we allow a single route in a solution to be represented by a convex combination of several routes, on the condition that every edge in E is selected binarily. Which is, what enables us to drop the binarity conditions on x_r, as shown in (8). We do not need to enforce $x_r \leq 1$ for all $r \in R^p$, because this is implied by constraints (2),(3) and (8).

3.4 Complete formulation

To prevent routes starting and ending at a charging station we link the partial routes to the trucks that will drive them and introduce an additional constraint. The p-step variable x_r will be rewritten as x_r^t, which will take a value > 0 if p-step r is driven by truck t. Next, we introduce variable y_t which is 1 if truck t is used. We also introduce parameter ϕ_r which is the recharge amount on p-step r. The complete formulation will then be,

$$\min \sum_{t \in T} \sum_{r \in R^p} d_r x_r^t \tag{10}$$

$$s.t. \sum_{t \in T} \sum_{r \in R^p} a_r^i x_r^t = 2 \qquad \forall i \in N' \tag{11}$$

$$\sum_{t \in T} \sum_{r \in R^p} e_r^i x_r^t = 0 \qquad \forall i \in N' \tag{12}$$

$$\sum_{t \in T} \sum_{r \in R^p} q_r^i x_r^t \geq 0 \qquad \forall i \in N' \tag{13}$$

$$\sum_{t \in T} \sum_{r \in R^p} b_{ij}^r x_r^t = \theta_{ij} \qquad \forall (i,j) \in E \tag{14}$$

$$\sum_{i \in N'} \sum_{r \in R^p} b_{0i}^r x_r^t + b_{in+1}^r x_r^t = 2y_t \qquad \forall t \in T \tag{15}$$

$$x_r^t \leq y_t \qquad \forall r \in R^p \; \forall t \in T \tag{16}$$

$$x_r^t \geq 0 \qquad \forall r \in R^p \; \forall t \in T \tag{17}$$

$$y_t \in \{0,1\} \qquad \forall t \in T \tag{18}$$

$$\theta_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{19}$$

The objective function (10) minimizes the total battery used over all trucks. Constraints (11) ensure all customers are visited. Constraints (12) ensure that the p-steps are connected by the same truck and constraints(13) will see to it that the power left over at the beginning of the current p-step is at most that of the preceding p-step. Constraints (14) remain relatively unchanged from (7). Then by adding constraints (15) and (16) we ensure that a complete route always starts en ends at the depot. If a truck t is selected to drive a route we have to select one p-step that leaves from the depot and one p-step that returns to the depot. Lastly, constraints

(17)-(19) define the bounds on the decision variables.

4 Solution Approach

In the next part of this thesis, we compute the LP bound of the p-step formulation with a branch-and-bound algorithm to solve the EVRP. This is to compare computation times and solution values of this formulation with different values of p.

Since the number of variables will increase when p increases, we make use of a column generation algorithm to compute the LP bounds. By using this algorithm we limit the number of variables included in the formulation, which yields something called a restricted master problem (RMP). The RMP is solved and in the next step by solving a pricing problem the next negative reduced cost variables are searched for. Hence, the algorithm will follow an iterative process. In each step, at least one negative reduced cost variable is added to the RMP. Unless no negative reduced costs can be found, in which case the solution to the current RMP is the optimal solution to the LP relaxation.

For our column generation algorithm, we will use formulation (10)-(19). Where in the RMP we will perform column generation on the p-step variables x. This means that we will not include all variables x in the RMP, but we will include all variables θ.

4.1 Pricing problem

The goal of the pricing problem is to identify a feasible p-step with the lowest possible cost. Take λ, μ, η and π as the dual variables corresponding to constraints (11), (12), (13) and (14) respectively. For convenience, we define $\lambda_l = \mu_l = \eta_l = 0$ for $l \in \{0, n+1\}$. Aside from the route and customer-related dual variables, we also have to take the dual variables corresponding to 15), into consideration. Since we add all p-steps to all trucks we take the most negative duals, which we refer to as γ, and add this to the fixed part to the reduced cost. Lastly, for constraints (16) we do not have a dual variable as the route does not exist yet so this would always be zero. The reduced cost $RC(r)$ of a given p-step (P_r, δ_r) such that s is the starting point and f is the ending point on P_r is.

$$RC(r) = \sum_{(i,j)\in P_r} d_{ij} + \gamma - \pi_{ij} - 2\lambda_j - \lambda_s + \lambda_f - \mu_s + \mu_f + \eta_s q_s - \eta_f q(r) \tag{20}$$

Since we defined the reduced costs for a fixed starting node s and final node f where $(s \neq f)$ of P_r, we consider so-called (s, f)-pricing problems (Dollevoet et al. 2020). Since we do not need to consider $n + 1$ starting node or 0 as a final node, we only construct $(n + 1)n$ pricing problems.

Note that for $\eta_s q_s - \eta_f q(r)$ we can rewrite the equation as $\eta_s \delta_r - \eta_f(\delta_r - \sum_{(i,j)\in P_r} q_i - q_j + \phi_j)$ which is the current leftover battery minus the charge needed to get to from i to j plus the recharge amount if j is a recharge station. To simplify, for a route r we can say $w_j = q_i - q_j + \phi_j$ and then $w(r) = \sum_{(i,j)\in P_r} w_j$ (note that in the case that we visit a charging station we can also increase the battery percentage). Thus we can rewrite our $RC(r)$ as,

$$RC(r) = \sum_{(i,j)\in P_r} d_{ij} + \gamma - \pi_{ij} - 2\lambda_j - \lambda_s + \lambda_f - \mu_s + \mu_f + \eta_s\delta_r - \eta_f(\delta_r - w(r)) \tag{21}$$

In the adjusted formulation, it is easy to see that δ_r is linear in $\eta_s - \eta_f$. Thus, if $\eta_f \geq \eta_s$ and the charging station is not visited it is optimal to set δ_r to the minimal value $w(r)$, then in the case $\eta_f < \eta_s$ and a charging station is not visited it is optimal to set δ_r to the maximal value, 100%. Lastly, if a charging station is visited it is always optimal to set δ_r to the minimal value. The reason that if $\eta_f < \eta_s$ and a charging station is visited it is still optimal to set δ_r to the minimal value is that the $q(r)$ is 'fixed'. For this, we refer back to Figure 3, regardless of the start minimum after visiting a recharging station the battery will be 100% thus the battery usage after visiting the station is fixed. The possible optimal values for δ_r result in two different scenarios to compute costs for each (s, f)-pricing problem.

For given values for δ_r, we can derive fixed costs C_{sf} and variable costs c_{ij} assigned to each arc $(i, j) \in A$. The computation of both costs depends on if the minimal or maximal value is assigned to the prior leftover battery δ_r. If the minimal value is assigned to δ_r we get fixed costs $C_{sf} = \gamma - \lambda_s + \lambda_f - \mu_s + \mu_f + (\eta_s - \eta_f)w_s$ and variable costs for each arc $c_{ij} = d_{ij} - \pi_{ij} - 2\lambda_j - \eta_f w_j$. Then if the maximal value is assigned to δ_r we get $C_{sf} = \gamma - \lambda_s + \lambda_f - \mu_s + \mu_f + (\eta_s - \eta_f) * 100\%$ as fixed costs and $c_{ij} = d_{ij} - \pi_{ij} - 2\lambda_j + \eta_s w_j$ as variable costs on the arcs.

Thus the sum of the variable costs per arc and the constant cost C_{sf} are the reduced cost of the variable associated with a path P_r and end battery percentage δ_r. A path is feasible if the battery percentage never exceeds 100% or drops below 0. Also if the starting point is at the depot, $s = 0$ the path consists of at most p arcs, otherwise, the path consists of exactly p arcs. Finding a feasible negative reduced cost can be done by solving an elementary shortest path problem with resource constraints (ESPPRC). The resource constraint lies in the amount of battery percentage left, for arc (i, j) a $q(r) - w_j < 0$ is not feasible.

4.2 ESPPRC

To solve our pricing problem we formulate it as an ESPPRC. The first step we took was to construct a standard integer programming problem. We again consider an undirected graph $G = (N, A)$. With $N = C \cup S \cup \{0, n + 1\}$ the set of nodes and A the set of arcs. Where the set $A^+(i)$ denotes all outgoing arcs from node i and $A^-(i)$ the incoming arcs. The decision variable x_{ij} is 1 if arc (i, j) is used and 0 otherwise. Variable y_{ij} represents the current available energy at j. If there is a recharge station on the route we use ϕ_j which represents the amount of battery that is recharged. Since the routes have not been made this means ϕ_j is a decision variable that can take any value between 0 and the total battery capacity Q. Parameter δ_{ij} is the battery usage on arc (i, j) and c_{ij} is the variable cost.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{22}$$

$$s.t. \sum_{(i,j)\in A^+(i)} x_{ij} - \sum_{(j,i)\in A^-(i)} x_{ji} = \begin{cases} 1 & if\ i = s \\ -1 & if\ i = f \\ 0 & otherwise \end{cases} \qquad \forall i \in V \tag{23}$$

$$\sum_{(i,j)\in A^+(i)} x_{ij} \leq 1 \qquad \forall i \in V \tag{24}$$

$$\sum_{(i,j)\in A} x_{ij} \leq p \tag{25}$$

$$y_{ti} - \delta_{ij}x_{ij} \geq Q(x_{ti} - 1) \qquad \forall t, i \in N\, \forall j \in N, t \neq i, i \neq j \tag{26}$$

$$y_{ti} - \delta_{ij}x_{ij} - y_{ij} \geq Q(2 - x_{ti} - x_{ij}) \qquad \forall t, i \in N \forall j \in N, t \neq i, i \neq j, i \neq s, i \neq f \tag{27}$$

$$y_{ti} - \delta_{ij}x_{ij} + \phi_j - y_{ij} \geq Q(2 - x_{ti} - x_{ij}) \qquad \forall t, i \in N \forall j \in S, t \neq i, i \neq j, i \neq s, i \neq f \tag{28}$$

$$0 \leq y_{ij} \leq Qx_{ij} \qquad \forall (i,j) \in A \tag{29}$$

$$\phi_j \geq 0 \qquad \forall j \in S \tag{30}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A \tag{31}$$

As opposed to the 'normal' elementary shortest path problem we added the constraints (25)-(30) to limit the arcs used for the p-step and to make sure we do not exceed the battery capacity. Constraints (26) guarantee we can reach the next node j from the current node i, if we use current node i. Constraints (27) ensure that if j is a customer or depot node, current available energy y_{ij} at j is equal to the previous available energy y_{ti} at i minus the battery used d_{ij}. Constraints (28) considers the case that j is a charging station, current available energy y_{ij} depends on the previous available energy y_{ti} at i minus the used battery δ_{ij} plus the amount of energy the vehicle recharged at station j. Constraints (29) enforce that the current available energy y_{ij} never exceeds the maximum capacity.

4.3 Labelling Algorithm

Since the MIP takes a lot of time to solve we will also make use of a bidirectional labelling algorithm. This algorithm will be based on the algorithm shown in Dollevoet et al. (2020) and Gschwind et al. (2018), but will have some minor modifications to take into account for the different requirements we have to meet. The general outline of how the labelling algorithm works is shown in Figure 4

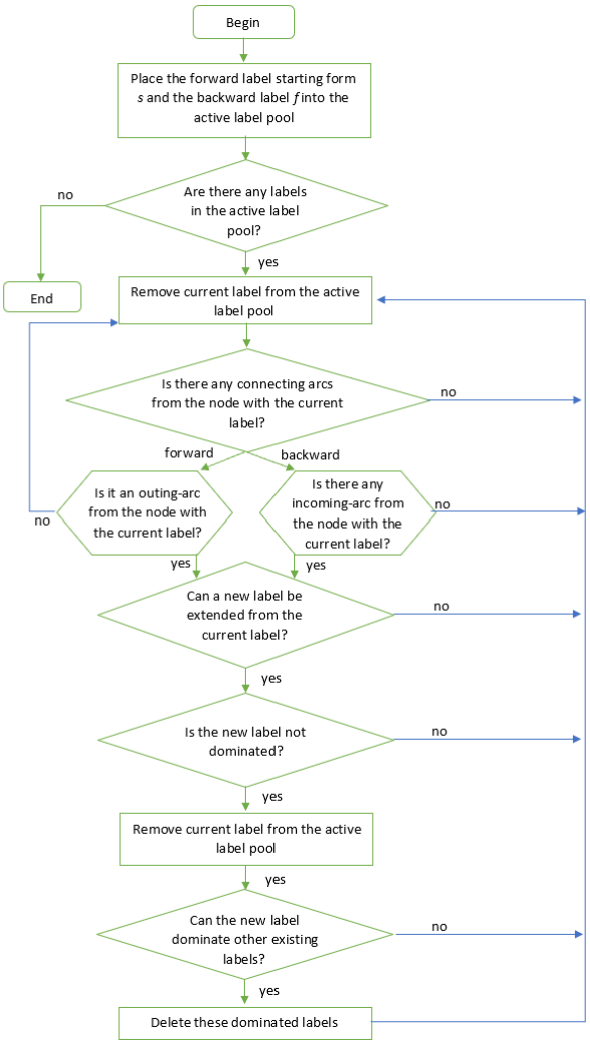

Figure 4: Illustration extension procedure in the labelling algorithm

A partial path corresponding to the label L is represented by P_L with as its attributes: the terminal node $N(L)$, number of arcs $A(L)$, the leftover battery $q(L)$ and the cumulative cost $c(L)$. Additionally, we will also use a vector $U(L) \in \mathbb{B}^{|N|}$ to indicate if a node can be reached. That means $U_i(L) = 1$ if we already visited node i or extending to node i leads to infeasibility. To be able to retrieve the path we took in our p-step we keep track of a vector $V(L)$ for which $V_i(L)$ indicates whether a node i has been visited. The initial forward label L which corresponds to a partial path starting at s is given by, $N(L) = s$, $A(L) = 0$, $q(L) = q_s$, $U_s(L) = 1$ and $U_i(L) = 0$ for all $i \in N \setminus \{s\}$. A partial path P_L corresponding to a forward label L can be extended along $(N(L), j) \in A$ to create a new label L' corresponding to a new partial path s through $N(L)$ to j.

To extend partial path P_L to $P_{L'}$ we use the following extension functions, where $F_i(L)$ is a function that takes the value 1 if extending to node i would lead to a violation in the battery capacity. For extension function $c(L')$ in the case we visit a charging station we 'fix' the battery percentages of the preceding nodes to their minimal value. In all other cases, the costs related

to battery percentage will be calculated when we merge a forward and backward label.

$$c(L') = c(L) + c_{N(L)j} \tag{32}$$

$$q(L') = q(L) - q_j \tag{33}$$

$$N(L') = j \tag{34}$$

$$A(L') = A(L) + 1 \tag{35}$$

$$U_i(L') = \begin{cases} 1 & if\ i = j \\ max\{U_i(L), F_i(L')\} & otherwise \end{cases} \qquad \forall i \in N \tag{36}$$

In analogy to the forward label a backward label L is extended in the reverse direction of an arc $(j, N(L)) \in A$. Since we want to combine forward and backward labels to construct feasible (s, f) paths we extend a forward label L for which $A(L) < p - \lfloor \frac{p-1}{2} \rfloor - 1$ along all arcs $(N(L), j) \in A$ such that $U_j(L) = 0$ and $j \neq f, j \neq n+1$. With the exception that L is extended to f in case s is the depot, because then a path may consist of less than p arcs. We extend a backward label L for which $A(L) < \lfloor \frac{p-1}{2} \rfloor$ in the reverse direction $(j, N(L)) \in A$ such that $U_j(L) = 0$ and $j \neq s, j \neq 0$. The limits imposed on $A(L)$ for extending the labels ensure that the longest partial path of both directions differs at most by one arc. Which results in roughly the same amount of forward and backward labels.

If there are no more labels in the active label pool, we start merging the forward and backward labels to construct full paths. We can find a feasible path from s to f consisting of exactly p arcs if we merge a forward label L with $A(L) = p - \lfloor \frac{p-1}{2} \rfloor - 1$ and a backward label L' for which $A(L') = \lfloor \frac{p-1}{2} \rfloor$ such that the merged path length is exactly p, when $(N(L), N(L')) \in A$, the battery capacity is not violated and $V(L)$ and $V(L')$ do not have any nodes in common. Where a merge is of L and L' defined as connecting partial path P_L with partial path $P_{L'}$. Where the merged path $P_{L+L'}$ visits nodes $V(L) \cup V(L')$ and has reduced costs $c(L + L') = c(L) + c(L') + c_{N(L)N(L')}$ However, when s is the depot we can consider all labels forward labels $A(L) \leq p - \lfloor \frac{p-1}{2} \rfloor - 1$ since the length of the partial path does not need to be exactly p. Note that if f is the depot we can do the same for backward labels which have $A(L) \leq \lfloor \frac{p-1}{2} \rfloor$.

A label L with ($N(L)$, $A(L)$, $c(L)$, $q(L)$, $U(L)$) dominates another label L' with ($N(L')$, $A(L')$, $c(L')$, $q(L')$, $U(L')$) if $N(L) = N(L'), q(L) \geq q(L'), U(L) \leq U(L'), c(L) \leq c(L')$ and $A(L) = A(L')$. This means that any feasible extension of the dominated label L' is also a feasible extension of the dominant label L. We perform a dominance check every time a new label is constructed by extension. Normally, L is checked for dominance over all other labels as well as all other labels against L. However, as explained further below, we can limit the number of dominance checks.

The labels will be stored in separate dynamic programming matrices one for forward labels and one for backward labels. The matrix will have n rows, corresponding to the nodes and $p + 1$ rows, to indicate the number of arcs. This means row i column j contains a list of all labels with i as its terminal node and with $A(L) = j - 1$. Also by using this way of storing labels we indirectly keep track of both $N(L)$ and $A(L)$. This also means they do not have to be checked explicitly for dominance as we only compare labels from the same matrix entry. We

will also order the labels based on reduced costs to limit the number of dominance checks we have to perform even further. This is because if the top label is dominated we will only save the dominant label and if it is not we do not need to check dominance for the other labels in the list. This in turn also helps with how many merges we have to attempt since we only want merges that result in a partial path with negative reduced costs.

5 Combined formulation

In Section 3 we showed what the p-step formulation would look like for a vehicle routing problem with electric vehicles. However, in most EVRPs there are also capacity constraints that need to be taken into account as the EVRP is most commonly seen as an extension of the CVRP problem. Thus in this section, we will give the combined p-step formulation of the CVRP as shown in Dollevoet et al. (2020) and our formulation as shown in Section 3.4. To achieve this we need to redefine our p-step first. Since we also need to take capacities into account we add v_r as an additional value which we need to remember per p-step, (P_r, δ_r, v_r). Where we define v_r as the cumulative demand of the customers on a route before arriving at the first location on P_r. Thus we refer to v_r as the prior demand of r and by $g(r)$ we denote the total demand of the customers that are visited on p-step r. Adding capacity to the p -steps results in an additional feasibility condition on the p-steps, namely $0 \leq v_r + g(r) \leq G$ where G stands for the truck capacity.

We also need to take into account that two p-steps r and s can be connected only if the demand of the first location of s is the same as that of the last location of r, that is $v_r + g(r) = v_s + g_i$ where g_i is the demand of location i. To represent this connectivity in our formulation we introduce g_r^i. Let g_r^i be equal to $v_r + g_i$ if i is the first location on the p-step, $-v_r - g(r)$ if i is the last location and 0 otherwise. If we formulate this as a constraint for our problem we get $\sum_{t \in T} \sum_{r \in R^p} g_r^i x_r^t = 0$ for all $i \in N'$. Similar to how it is explained in Dollevoet et al. (2020) we have a finite amount of options if we formulate each p-step as a linear combination of the maximal and minimal demand together with the maximal and minimal prior leftover battery needed for each p-step. Thus we have four combinations of extremes; I) minimal battery and minimal demand, II) minimal battery and maximal demand, III) maximal battery and minimal demand and IV) maximal battery and maximal demand. Thus, we can relax these equalities to inequalities without the p-step formulation becoming invalid. So, we need to add

$$\sum_{t \in T} \sum_{r \in R^p} g_r^i x_r^t \geq 0 \forall i \in N' \tag{37}$$

to our formulation. For the complete mathematical formulation, we refer to Appendix A.

5.1 Combined pricing problem

Since Equation (37) is added to the main problem we also need to incorporate it into the pricing problem. Let ϕ be the dual corresponding to this constraint. Here as well, for convenience, we define $\phi_l = 0$ for $l \in \{0, n+1\}$. Thus reduced costs $RC(r)$ of a given p-step (P_r, δ_r, v_r) such

that s is the starting point and f is the ending point of P_r is given by,

$$RC(r) = \sum_{(i,j)\in P_r} d_{ij} + \gamma - \pi_{ij} - 2\lambda_j - \lambda_s + \lambda_f - \mu_s + \mu_f + \eta_s\delta_r - \eta_f(\delta_r - w(r)) \\ - \phi_s(v_r + g_s) + \phi_f(v_r + g(r)) \tag{38}$$

Similar to the duals for battery percentage, we can recognise that $\phi_f - \phi_s$ is linear in prior demand v_r. Hence, we let prior demand v_r be the minimum value, 0, if $\phi_f \geq \phi_s$ otherwise, we assign the maximum value $G - D(R)$.

These reduced costs can be decomposed into multiple scenarios of (s, f)-pricing problems similar to what we did in Section 4.1 and like they did in Dollevoet et al. (2020). We can break the reduced cost for each (s, f)- pricing problem up into fixed costs and variable costs that are influenced by which arcs are used. Both of these are dependent on if the maximum or minimum battery percentage is assigned as explained in Section 4.1. In this case, it will also depend on if the minimal or maximal value is assigned to prior demand.

First, we look at the case that the minimum value is assigned to δ_r, if we also assign the minimum value to prior demand v_r we get:

$$C_{sf} = \gamma - \lambda_s + \lambda_f - \mu_s + \mu_f + (\eta_s - \eta_f)w_s + (\phi_f - \phi_s)g_s$$
$$c_{ij} = d_{ij} - \pi_{ij} - 2\lambda_j - \eta_f w_j + \phi_f g_j$$

Then if we assign the maximum value to v_r we get:

$$C_{sf} = \gamma - \lambda_s + \lambda_f - \mu_s + \mu_f + (\eta_s - \eta_f)w_s + (\phi_f - \phi_s)G$$
$$c_{ij} = d_{ij} - \pi_{ij} - 2\lambda_j - \eta_f w_j - \phi_s g_j$$

Second, we look at the case in which we assign the maximum value to δ_r, if we assign the minimum value to prior demand v_r we get:

$$C_{sf} = \gamma - \lambda_s + \lambda_f - \mu_s + \mu_f + (\eta_s - \eta_f)100\% + (\phi_f - \phi_s)g_s$$
$$c_{ij} = d_{ij} - \pi_{ij} - 2\lambda_j + \eta_s w_j + \phi_f g_j$$

Then if we assign the maximum value to v_r we get:

$$C_{sf} = \gamma - \lambda_s + \lambda_f - \mu_s + \mu_f + (\eta_s - \eta_f)100\% + (\phi_f - \phi_s)G$$
$$c_{ij} = d_{ij} - \pi_{ij} - 2\lambda_j + \eta_s w_j - \phi_s g_j$$

Which results in the (s, f)-pricing problem having four different scenarios each.

5.2 Modification to the labelling algorithm

The new pricing problem can still be solved as an ESPPRC with a labelling algorithm. If we store an additional value in our label for capacity we can use the labelling algorithm given in Section 4.3. To illustrate; a partial path corresponding to label L is represented by $P_L = \{N(L), A(L), q(L), c(L), U(L), V(L), g(L)\}$ where $g(L)$ represents the cumulative demand and

all other attributes remain as previously defined in Section 4.3. Next, we also need to add the extension function

$$g(L') = g(L) + g_j \tag{39}$$

to extend the partial path P_L to $P_{L'}$. Lastly, we would also need to update our dominance rule by including the condition that if a label L dominates L', $g(L) \leq g(L')$ must hold. The rest of the labelling algorithm will remain unchanged.

6 Computational results

This section presents and discusses the computational results of the p-step formulation. In Section 6.1 we show the LP bounds of the p-step formulation for a few values of p and if the bound could be obtained through enumeration within the time limit we show these computation times. In this subsection we also look at the relationship between instance size, number of p-steps and the size of p. Next in Section 6.2 we show how close the LP bounds of the smaller instances perform when compared to the optimal solution. Then in Section 6.3 we compare the performance, measured in time, of our two approaches to the pricing problem. Lastly, in Section 6.4 we show the LP bounds of the combined p-step formulation for a few instances

To obtain these results we use some benchmark instances that were introduced in Schneider et al. (2014) for computational tests. These data sets were created by using the most commonly used Solomon data sets. The data sets are categorized into three different types. First, there is Clustered (c), c101C10. Second, there is Random (r), r102C10. Lastly, there is RandomClustered (RC), rc102C10. The smaller instances are also indicated by the number of customers in the data set, for example, C10 means 10 customers. All data sets contain the locations of the customers, their demands and they also contain time windows and the corresponding service time. Furthermore, the instances contain the range of the vehicles, the capacity, the refuelling rate and indicate whether a node is a charging station, a customer or the depot. The datasets also contain fuel consumption rate however this is 1.0 in all data sets thus we take it as one unit of fuel equals one unit of distance. In the first few experiments, we will only use the location and range of the trucks from these data sets and in Section 6.4 we will also use the demand and capacity given.

Our algorithms are implemented in Java using the IDE Eclipse 2020-06 (version 2.3.200) and are executed on a PC with 3 GHz Intel Core i5-8500 processor and 16 GB RAM. We use CPLEX 20.1 with default settings to solve the RMP at each iteration of the column generation algorithm.

6.1 Master Problem

We show the LP bounds of the p-step formulation as shown in Section 3.4 we find in Table 2. We have included a few of the smaller problems as this allows us to compute the LP bounds within a reasonable time. We decided to impose a time limit of one hour, 3600 seconds. To compute the LP bounds we ran both the complete problem with enumeration over all feasible p-steps and the column generation algorithm, which give the same bounds if both are found

within the time frame. If we did not find an LP bound within this time frame the algorithm was stopped and we denote this with a − in the table.

Table 2: LP solutions

	p					
Instance	1	2	3	4	5	n+1
c101C5	167.746	175.173	178.775	200.169	200.169	200.169
c103C5	136.431	136.965	138.751	138.751	139.163	139.163
c206C5	164.556	165.977	174.31	191.645	201.164	201.164
r105C5	117.617	118.825	123.924	134.946	134.946	134.946
r202C5	114.459	114.459	114.459	114.459	114.459	114.459
r203C5	151.205	151.521	153.040	174.369	174.369	174.369
rc105C5	150.201	151.201	163.59	165.668	165.668	165.668
rc108C5	155.553	156.528	174.099	206.94	216.667	216.667
rc204C5	97.215	97.431	103.115	126.124	131.853	131.853
c101C10	200.132	211.980	204.921	212.816	220.716	−
c104C10	215.866	219.449	224.252	235.820	240.977	241.36
c202C10	162.085	163.675	176.635	175.827	178.254	226.091
c205C10	192.875	193.844	199.766	207.692	217.987	219.078
r102C10	171.396	172.491	174.034	179.458	187.005	227.078
r103C10	120.824	122.537	121.800	125.676	127.627	−
r203C10	165.420	165.682	178.034	181.958	181.115	196.097
rc102C10	326.748	339.811	358.211	410.489	414.549	415.839
rc108C10	253.555	254.839	306.133	324.799	324.846	335.507
rc201C10	195.335	197.420	207.265	221.375	236.965	261.666
rc205C10	247.451	255.350	256.002	290.304	317.112	324.758
c103C15	209.559	210.296	211.062	221.439	228.0544	−
c106C15	203.524	223.838	223.838	223.838	−	−
c202C15	278.244	278.481	285.059	287.123	316.429	343.312
c208C15	215.756	215.971	220.202	226.620	235.623	262.795
r102C15	234.691	259.227	259.227	263.107	271.288	−
r105C15	192.027	193.089	194.648	195.429	202.683	233.923
r202C15	226.812	228.251	227.062	228.949	236.326	−
r209C15	218.196	218.650	218.372	218.650	246.972	252.008
rc103C15	209.754	210.287	236.803	237.418	260.709	263.288
rc108C15	243.493	247.553	256.318	272.653	281.539	333.683
rc202C15	224.353	231.037	241.625	274.961	279.812	327.804
rc204C15	220.651	220.940	243.183	−	−	−

The first thing we note in Table 2 is that some instances do not allow our algorithms to determine the LP bounds within the time constraint. This is why we only choose to use the cases for which we can compute the LP bounds within the time frame in the remaining experiments. Then from the results in the Table 2 we can see that for every instance, the LP bound for a certain p is always lower than or equal to the LP bound of the same instance for an integer multiple of p. We find that for all instances with five customers, $C = 5$, that the LP bound is monotonically non-decreasing. Then in most instances with ten or fifteen customers, eight out of eleven and nine out of twelve respectively, we find it is even monotonically increasing when p increases.

This is, what we expected to find since it was found to be the case for the Capacitated Vehicle Routing Problem in the literature (Dollevoet et al. 2020). From this, we can conclude that our EVRP p-steps formulation acts in a similar fashion to the CVRP with p-steps.

In Table 3 we also show the computation times in seconds needed to construct the LP bounds of the p-step formulation if we do not use column generation. We only show the computation times for $p = 1, ..., 5$ as for $n + 1$ we were unable to construct the LP values within reasonable time for $C = 10$ and $C = 15$. We show this to both draw a better comparison with how much our column generation algorithm can improve the computation time and to further illustrate the trade-off between Set Partitioning and Arc-flow when it comes to getting the LP bounds.

Table 3: Computational time in seconds of enumeration

	p				
Instance	1	2	3	4	5
c101C5	0.293	0.232	0.208	0.416	0.255
c103C5	0.181	0.241	0.428	0.344	0.312
c206C5	0.192	0.222	0.273	0.25	0.25
r105C5	0.194	0.321	0.264	0.433	0.267
r202C5	0.155	0.242	0.282	0.286	0.296
r203C5	0.203	0.479	0.489	0.568	0.598
rc105C5	0.185	0.431	0.543	0.370	0.381
rc108C5	0.211	0.304	0.286	0.286	0.298
rc204C5	0.281	0.384	0.597	0.530	0.466
c101C10	0.674	2.483	18.355	72.347	137.629
c104C10	0.496	1.833	5.845	16.558	34.773
c202C10	0.667	3.132	28.582	92.93	85.716
c205C10	0.455	1.611	5.532	16.454	34.282
r102C10	0.577	1.542	3.43	12.113	22.313
r103C10	0.429	2.476	36.491	254.13	380.559
r203C10	0.542	3.15	14.346	28.673	40.109
rc102C10	0.434	1.714	3.544	5.058	12.238
rc108C10	0.452	2.117	2.255	3.375	9.424
rc201C10	0.696	4.386	12.019	15.98	20.884
rc205C10	0.409	1.416	4.784	8.146	12.294
c103C15	1.774	25.692	439.696	1718.289	–
c106C15	1.200	40.018	1024.214	–	–
c202C15	2.099	43.650	150.254	169.440	289.908
c208C15	1.477	18.732	159.039	631.405	737.775
r102C15	4.848	288.901	2418.978	–	–
r105C15	2.134	56.939	416.859	506.296	680.099
r202C15	2.556	46.722	441.004	1322.243	–
r209C15	2.414	53.585	418.221	819.277	1392.94
rc103C15	1.383	36.326	216.183	287.305	382.603
rc108C15	3.508	46.442	149.096	178.291	218.923
rc202C15	3.203	39.021	197.364	252.630	370.073
rc204C15	2.492	109.989	1489.42	–	–

From Table 3 we find that for a higher number of customers that have to be visited, the run-time increases drastically when p increases. In most cases with fifteen customers, it already

becomes impossible to solve through enumeration within a reasonable amount of time.

Since we did not include capacity or time windows in this problem it is hard to draw an exact comparison between our findings and those of Schneider et al. (2014) because our problem allows more solutions since there are fewer restrictions on the p-steps. However, since our solution values are slightly below their finding we believe our algorithm gives decent bounds for the problem without time window and capacity constraints.

We also looked at the possibility that how fast the algorithm ran could be related to the number of feasible p-steps. Because, even though they have the same number of customers, they might not have the same number of charging stations thus the number of feasible p-steps does not have to be equal. Furthermore, the number of feasible p-steps does not have to be equal among instances of the same size as this is heavily dependent on the locations of the customers.

In Figure 5 we plot the relationship between solution time and the number of p-steps for varying sizes of p when there are ten customers. The horizontal axis corresponds to the solution time and the vertical axis to the number of p steps. Where $p = 1$ is denoted by the circles, $p = 2$ by the triangles and $p = 3$ by the squares. From this figure, we can see that the amount of p-steps increases with p. We find that for $p = 1$ and $p = 2$ both the solution times and the number of feasible p-steps are approximately the same. For some instances of $p = 3$ this also seems to be the case but there is a larger deviation in number of feasible p-steps.

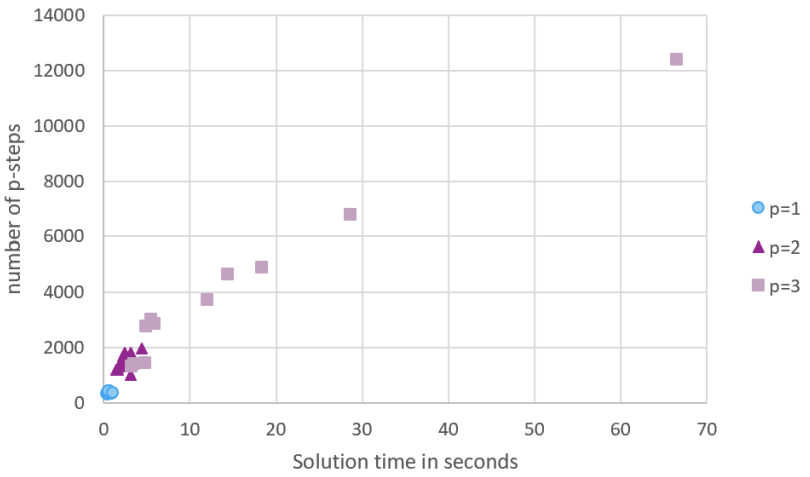

Figure 5: CPU compared to the number of p-steps for $C = 10$

From Figure 5 we can clearly see that when the number of feasible p-steps exceeds a certain point, approximately 3800, the run-time starts to increase more drastically. We find that one of the instances has more than two times the number of feasible p-steps as all other instances. This confirms our statement that because of the charging stations and different distances, the solution times are heavily dependent on the specific instance when it comes to solution time.

To further explore this we also looked at different instance sizes with the same value for p. In Figure 6 we plot the relationship between solution time and the number of feasible p-steps for $p = 1$. Where $C = 10$ is denoted by the circles, $C = 5$ by the triangles and $C = 15$ by the squares.

From this figure, we can see that the computation time seems to depend more on the size of

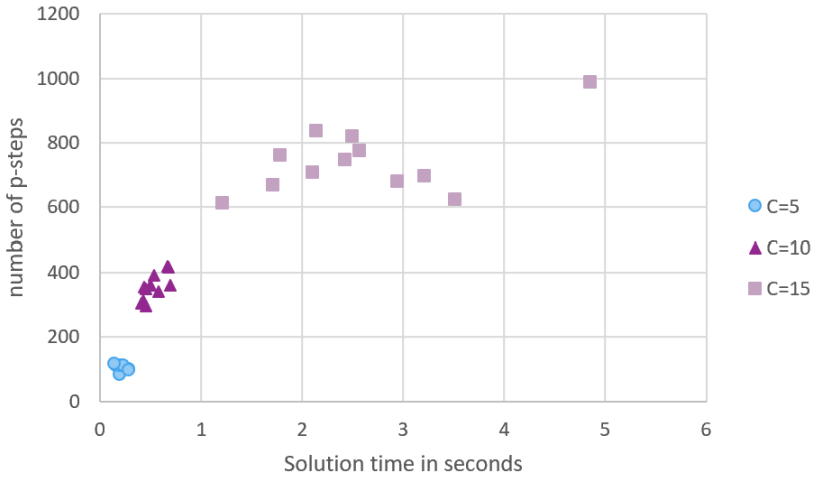

Figure 6: CPU compared to the number of feasible p-steps for $p = 1$

the instance than on how many partial routes we consider. As we saw when looking at Figure 5, we see that for p is one most instances have similar size and computation times. We only see that if the set of allowed p-steps is much larger it can impact the computation time for the observation for $C = 15$ at approximately 5.8 seconds.

Next, we look at the case that $p = 2$ in Figure 7. From this figure, we can again see that the number of customers has a bigger influence on computation time. Like in Figure 6 we see that for $C = 5$ and $C = 10$ the computation times and the number of p-steps are kind of clustered. All instances with five customers are solved in 0.15 to 0.4 seconds and the instances with ten customers take between 1.1 and 4.4 seconds. However, when we look at $C = 15$ we find an outlier that again takes at least twice as long to solve than all other instances. Which even further confirms the statement that the performance of the algorithm is dependent on the instance.

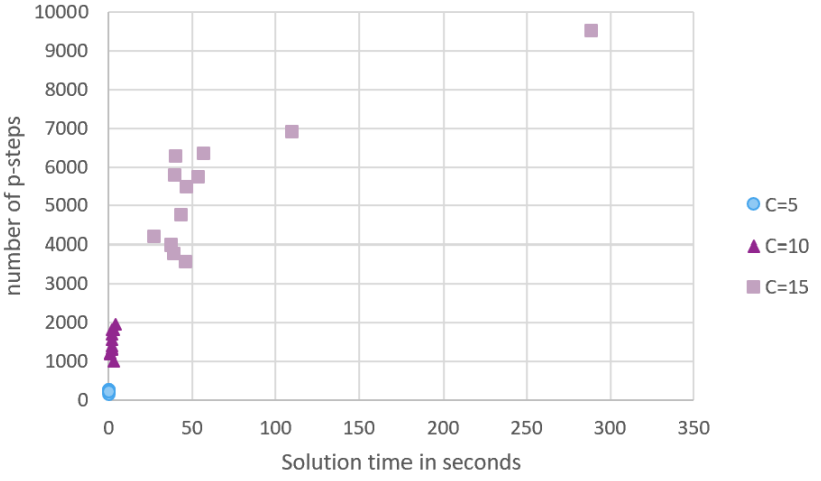

Figure 7: CPU compared to the number of feasible p-steps for $p = 2$

What we can also see from both of these figures is how much the computation time can blow up for larger instances and how many more p-steps we have to look at when p increases.

6.2 Performance LP bounds

Since if we generate all p-steps and then solve for the LP relaxation for instances with five customers our model solves in a matter of seconds we decided to solve the problem to integer optimality. In Figure 8 we show the relative gap between the LP bound and the optimal value for some instances with five customers for different p-steps. On the vertical axis, we show the gap in percentages and on the horizontal axis, we show the corresponding size of p. Then in the legend we show which line corresponds to which problem instance with $C = 5$ customers.

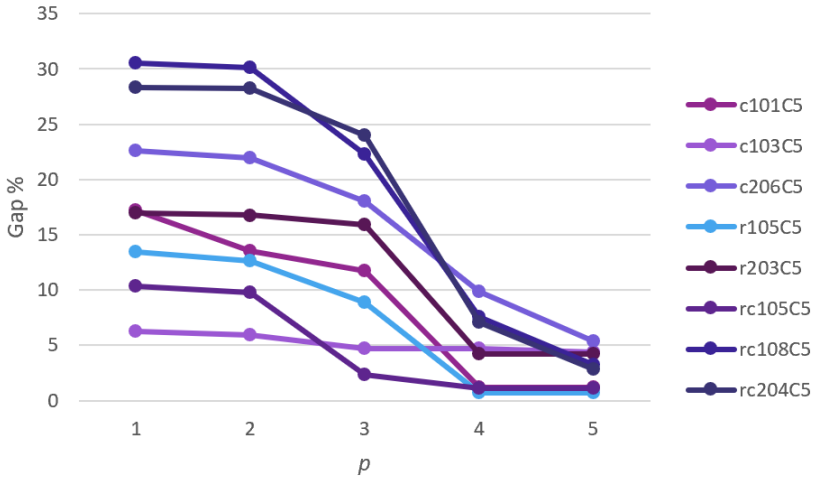

Figure 8: Gap in % LP and optimal value

As we can see from the results in Figure 8 we can establish that the set partitioning formulation ($p = 1$) gives the worst bound when compared to larger values of p. We also see that in 6 out of the 8 cases, we look at we see the biggest improvement when we go from $p = 3$ to $p = 4$. This indicates that using p-steps can result in a much tighter bound without having to resort to using the set-partitioning formulation i.e. $p = n + 1$ which is not shown in this figure as all instances for $C = 5$ had the same value for $p = 5$ and $p = 6$.

6.3 Performance of the pricing problems

For the restricted master problem we started with not all possible p-steps as variables but only a small, but feasible, set of p-steps that were constructed in a greedy way. As p increases the computational times also significantly increase. We tried to counter this by using column generation, for which we made two methods of generating said columns. Next, we illustrate the computational performance of these two methods on the benchmark instances that we were able to solve within our time limit, i.e. 3600 seconds.

When computing the LP bounds with column generation we find two opposing effects that influence the computation time of the pricing problems. The first effect we find is that as p increases it causes the worst-case number of computations of the labelling algorithm to grow exponentially. Larger values of p will also have a similar effect on the MIP pricing problem. Since there is a bound on the length of the path we solve the MIP for, if p is small there are fewer possible ways to construct paths of length p. While if p is larger we need the worst-case

number of computations will increase. The second effect we find is that when p increases the number of (s, f)-pricing problems decreases from quadratic in customers, n^2, to one. However as the number of iterations until convergence influences the speed of the column generation algorithm the computation times do not necessarily have to increase when p increases.

Table 4 provides the computational times in seconds of the instances, which names are provided in the first column. The second column denotes which pricing problem we used to solve the instance. Where MIP means we used the exact formulating of the ESPPRC denoted in Section 4.2 by implementing it in Java and CPLEX. And labelling means that we used the labelling algorithm as described in Section 4.3. The last six columns on the first row indicate which size for p was chosen.

Table 4: Time comparison pricing algorithms in seconds

		p				
Instance		2	3	4	5	n+1
c104C10	MIP	22.566	28.564	32.003	39.707	520.233
	Labelling	0.357	2.910	7.239	9.785	290.458
c202C10	MIP	11.552	13.871	16.064	22.373	420.233
	Labelling	1.654	1.997	6.636	10.969	254.273
c205C10	MIP	23.683	29.499	37.157	34.608	441.969
	Labelling	0.394	1.977	5.702	11.547	199.943
r102C10	MIP	10.873	11.742	12.63	11.073	455.191
	Labelling	1.101	1.793	7.567	8.434	310.047
r203C10	MIP	14.476	13.852	17.636	12.777	302.011
	Labelling	1.815	2.704	6.389	9.721	187.538
rc102C10	MIP	5.637	6.672	9.821	12.284	290.702
	Labelling	1.088	1.873	6.761	9.691	120.028
rc108C10	MIP	7.981	8.252	8.464	8.603	135.783
	Labelling	1.136	1.202	2.775	3.137	45.839
rc201C10	MIP	10.698	11.156	11. 507	11.532	134.658
	Labelling	5.055	5.497	6.339	7.111	87.398
rc205C10	MIP	5.007	5.544	5.657	5.759	64.627
	Labelling	1.332	1.845	2.425	3.658	48.494
c202C15	MIP	19.795	21.511	124.374	233.864	1535.375
	Labelling	2.976	3.258	27.431	111.266	923.111
c208C15	MIP	19.744	20.570	127.902	330.142	2543.286
	Labelling	1.881	3.129	7.124	84.817	1297.696
r105C15	MIP	18.379	20.966	58.539	84.242	1928.233
	Labelling	2.406	5.445	39.787	44.140	688.433
r209C15	MIP	18.142	20.466	64.804	227.351	677.042
	Labelling	2.991	4.140	28.764	110.547	320.311
rc103C15	MIP	12.054	27.387	56.339	139.701	463.008
	Labelling	6.586	19.251	33.127	79.297	237.202
rc108C15	MIP	20.054	23.996	45.495	124.518	382.157
	Labelling	1.257	6.965	20.073	63.260	236.529
rc202C15	MIP	16.455	18.640	20.858	223.034	888.919
	Labelling	2.007	2.527	8.919	110.153	462.656

From Table 4 we find that generally, the computational times for the pricing problem are

faster when we use the labelling algorithm compared to the MIP algorithm. This, however, is probably the result of the fact that the MIP algorithm only adds one p-step per iteration while the labelling algorithm adds all p-steps that have negative reduced costs. Keeping this in mind we find that both approaches are successful in speeding up the algorithm when we compare it to the case that we just enumerate over all p-steps. We find that once $p \geq 3$ the column generation algorithm significantly speeds up the algorithm as without it we were not always able to find the LP bound within the time frame. Thus we show the potential gains that can be obtained from using column generation.

6.4 Computation LP solutions combined formulation

Lastly, we want to show the computation times of the LP bounds for the combined formulation. For this, we use some of the same instances we used in our previous computational experiments, this time including the demand per customer. Table 5 shows the LP bounds of the p-step formulation for a selection of instances that visit fifteen customers. We included some of the same instances we used in our experiments in Section 6.3, as indicated by the first column. We computed the bounds for low values of p, namely $p = 2, 3, 4, 6$ as indicated by columns 2 through 5. Then in the last column, we show the best bound found in the literature (Kucukoglu et al. 2021; Schneider et al. 2014) (note: these bounds do include time windows).

Table 5: LP solutions combined model

	p				
Instance	2	3	4	6	best bound
c202C15	296.975	298.649	302.074	341.024	383.610
c208C15	250.837	251.121	259.312	297.436	300.550
r105C15	227.785	229.279	232.024	234.513	336.150
r209C15	245.683	246.301	257.619	271.51	358.00
rc103C15	252.591	251.806	265.345	299.167	397.67
rc108C15	292.102	299.952	292.807	320.291	370.25
rc202C15	282.46	276.91	287.748	372.639	394.39

We find that the combined problem is also strictly increasing in p for most of these instances. We also see that multiples of p are always larger or equal, i.e. the LP bound for $p = 2$ is smaller or equal to that of $p = 4$, $z_2 \leq z_4$. Since all solutions were found within a reasonable time, less than 3600 seconds, we can say that there is some merit to be found in using a p-step formulation for the EVRP which also has capacity constraints as well.

To illustrate the impact the additional conditions of demands and capacity has on the run time we plotted the difference in Figure 9. Where on the horizontal axis we show the size of p and on the vertical axis the increase in computational time in seconds. We see that for most instances, five out of seven, the runtime will increase quite evenly with p. These increases all seem to be reasonable with the problem size. However, for two of the instances, we note a far steeper increase. This brings us back to our previously mentioned point of how dependent the performance is on each individual instance.

With the results shown in Figure 9 we further want to highlight that improving the efficiency

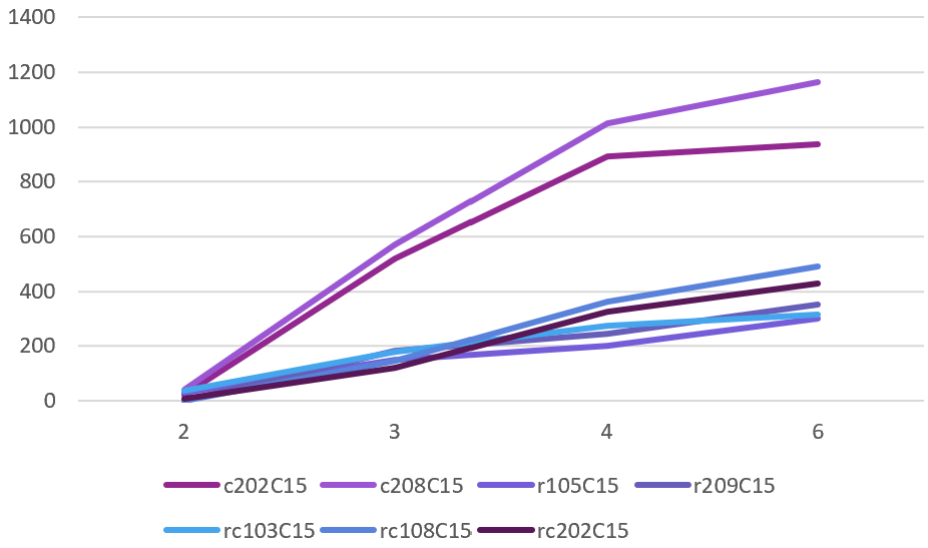

Figure 9: Increase in computation times in seconds

of the pricing algorithms or even using heuristics can save more time for these problems that are more complex.

7 Conclusion

This thesis covers the formulation of the p-step problem for the Electric Vehicle Routing problem. This formulation can be seen as a generalised version of both the arc-flow and set partitioning formulation. We constructed a restricted master problem and pricing problems to solve this p-step formulation with column generation.

The introduction of electric vehicles to the vehicle routing problem had one major challenge, which was the limited reach due to battery restrictions and the possibility of expanding the reach by visiting a charging station. To limit this restriction we also considered the possibility of visiting recharge stations on our route. Which came with a minor challenge of making sure the vehicles returned to the depot and not the recharge station. In further research assigning p-steps to vehicles could be done more efficiently as the way we currently do it generates a lot of unnecessary variables which slows down the algorithm.

We partially solved the slowness of the algorithm by using column generation. We found that the pricing problem can be solved in a significantly shorter time, both if we use a labelling algorithm and if we use an exact solver. But in further research, these methods could also be constructed in a more efficient way. As an example, the labelling algorithm could be modified to solve the problem in a heuristic way by not saving all labels but only one. Once this heuristic does not find any reduced costs we would revert back to the exact labelling algorithm. Or one could explore other heuristic-based methods. Furthermore, while tackling pricing problems, various strategies such as k-cycle elimination or ng-route relaxation may be used to speed up the process.

Nevertheless, we found evidence of potential advantages in using p-steps as the bounds seem to form a tighter gap on the optimal value as p increases even if computation time also slightly

increases. These potential advantages are especially noteworthy for the larger instances where the difference in the solution value for some values of p is very small compared to the set partitioning bounds that are found.

We also took the first step towards extending the formulation by incorporating capacity constraints. Although we found this formulation to be more complex, we still found that in this case there is also merit to be obtained from using the p-step formulation as we can compute decent bounds within a reasonable amount of time. Furthermore, like with the p-step formulation that only takes the driving distance and recharges into account, it is possible to further improve the solution methods by implementing more of the current state-of-the-art algorithms.

Since this thesis covers the p-step formulation for electric vehicles the formulation could be extended to create more complete vehicle routing problems. For example by including time windows to make it an EVRPTW. Furthermore, if we one was to include time windows into the problem we could also look at a non-linear charging function and different types of charging stations, e.g. fast-charging and normal charging stations. Additionally, one could look into more efficient ways of solving the p-step formulation.

References

Bruglieri, M., Mancini, S., Pezzella, F., & Pisacane, O. (2019). A path-based solution approach for the green vehicle routing problem. *Computers & Operations Research*, *103*, 109–122.

Dollevoet, T., Munari, P., & Spliet, R. (2020). *A p-step formulation for the capacitated vehicle routing problem* (tech. rep.).

EEA. (2021). Annual european union greenhouse gas inventory 1990-2019 and inventory report 2021 [Accessed: 2022-05-22].

Erdelić, T., & Carić, T. (2019). A survey on the electric vehicle routing problem: Variants and solution approaches. *Journal of Advanced Transportation*, *2019*.

Erdoğan, S., & Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, *48*(1), 100–114.

Feng, W., & Figliozzi, M. (2013). An economic and technological analysis of the key factors affecting the competitiveness of electric commercial vehicles: A case study from the usa market. *Transportation Research Part C: Emerging Technologies*, *26*, 135–145. https://doi.org/https://doi.org/10.1016/j.trc.2012.06.007

Gschwind, T., Irnich, S., Rothenbächer, A.-K., & Tilk, C. (2018). Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. *European Journal of Operational Research*, *266*(2), 521–530.

Koç, Ç., & Karaoglan, I. (2016). The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing*, *39*, 154–164.

Kucukoglu, I., Dewil, R., & Cattrysse, D. (2021). The electric vehicle routing problem and its variations: A literature review. *Computers & Industrial Engineering*, *161*, 107650.

Lin, J., Zhou, W., & Wolfson, O. (2016). Electric vehicle routing problem [Tenth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain]. *Transportation Research Procedia*, *12*, 508–521. https://doi.org/https://doi.org/10.1016/j.trpro.2016.02.007

Mancini, F. (2013). High resolution time-to-depth conversion using 3d grid tomography. *ASEG Extended Abstracts*, *2013*(1), 1–4. https://doi.org/10.1071/ASEG2013ab305

Mancini, S. (2013). *Multi-echelon freight distribution systems: A smart and innovative tool for increasing logistic operations efficiency* [Cited by: 3]. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84881004013&partnerID=40&md5=5aa06bfbcf6bef16add3d85919eae8f7

Mancini, S. (2017). The hybrid vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, *78*, 1–12.

Pourazarm, S., Cassandras, C. G., & Malikopoulos, A. (2014). Optimal routing of electric vehicles in networks with charging nodes: A dynamic programming approach. *2014 IEEE International Electric Vehicle Conference (IEVC)*, 1–7.

Rabbie, J. (2018). *A pulse algorithm for column generation for a generalized vehicle routing formulation* (Master's thesis). Erasmus University Rotterdam.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, *48*(4), 500–520.

Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications*. SIAM.

Wen, M., Linde, E., Ropke, S., Mirchandani, P., & Larsen, A. (2016). An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, *76*, 73–83.

Young, K., Wang, C., Wang, L. Y., & Strunz, K. (2013). Electric vehicle battery technologies. *Electric vehicle integration into modern power networks* (pp. 15–56). Springer.

Zhang, S., Gajpal, Y., & Appadoo, S. (2018). A meta-heuristic for capacitated green vehicle routing problem. *Annals of Operations Research*, *269*(1), 753–771.

Zhang, S., Gajpal, Y., Appadoo, S., & Abdulkader, M. (2018). Electric vehicle routing problem with recharging stations for minimizing energy consumption. *International Journal of Production Economics*, *203*, 404–413.

A Combined formulation: mathematical notation

The complete combined mathematical formulation is given by:

$$\min \sum_{t\in T}\sum_{r\in R^p} d_r x_r^t \tag{40}$$

$$s.t. \sum_{t\in T}\sum_{r\in R^p} a_r^i x_r^t = 2 \qquad \forall i \in N' \tag{41}$$

$$\sum_{t\in T}\sum_{r\in R^p} e_r^i x_r^t = 0 \qquad \forall i \in N' \tag{42}$$

$$\sum_{t\in T}\sum_{r\in R^p} q_r^i x_r^t \geq 0 \qquad \forall i \in N' \tag{43}$$

$$\sum_{t\in T}\sum_{r\in R^p} g_r^i x_r \qquad t \geq 0 \forall i \in N' \tag{44}$$

$$\sum_{t\in T}\sum_{r\in R^p} b_{ij}^r x_r^t = \theta_{ij} \qquad \forall (i,j) \in E \tag{45}$$

$$\sum_{i\in N'}\sum_{r\in R^p} b_{0i}^r x_r^t + b_{in+1}^r x_r^t = 2y_t \qquad \forall t \in T \tag{46}$$

$$x_r^t \leq y_t \qquad \forall r \in R^p \; \forall t \in T \tag{47}$$

$$x_r^t \geq 0 \qquad \forall r \in R^p \; \forall t \in T \tag{48}$$

$$y_t \in \{0,1\} \qquad \forall t \in T \tag{49}$$

$$\theta_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{50}$$