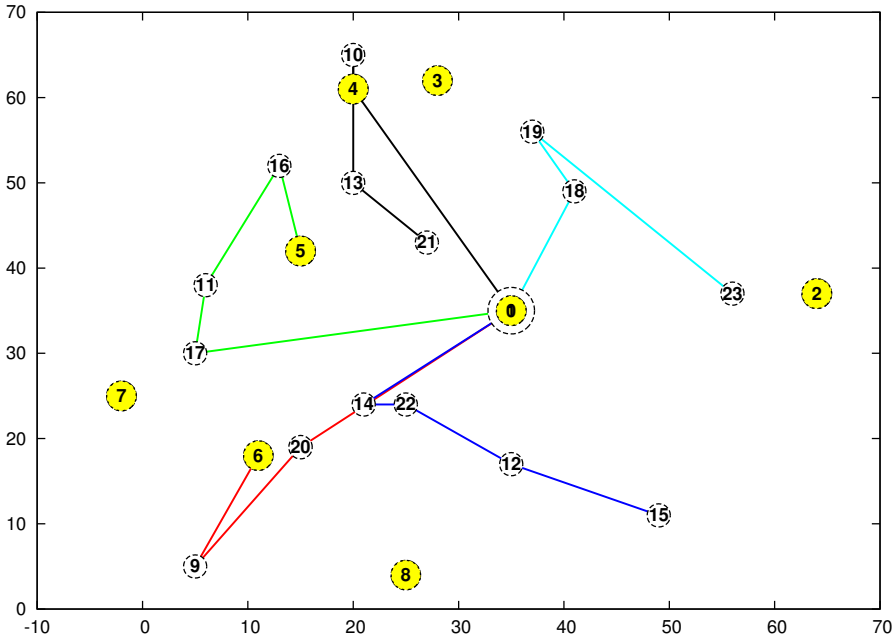

Optimization of routes for a fleet of plug-in hybrid vehicles

Mathematical modeling and solution procedures

Master's thesis in Engineering Mathematics and Computational Science

Jonathan Ruffieux

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Master's thesis 2016

Optimization of routes for a fleet of plug-in hybrid vehicles

Mathematical modeling and solution procedures

JONATHAN RUFFIEUX

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2016

Optimization of routes for a fleet of plug-in hybrid vehicles
Mathematical modeling and solution procedures.

Supervisor: Peter Lindroth, Volvo Group Trucks Technology
Examiner: Ann-Brith Strömberg, Department of Mathematical Sciences

Master's Thesis 2016
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A visualization of an optimal routing schedule for a fleet of five hybrid vehicles.

Typeset in LaTeX
Gothenburg, Sweden 2016

Optimization of routes for a fleet of plug-in hybrid vehicles
Mathematical modeling and solution
JONATHAN RUFFIEUX
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

We have developed mathematical models and optimization methods for the problem of routing a fleet of plug-in hybrid vehicles. This problem is referred to as the hybrid vehicle routing problem (hybrid VRP) and it is a generalization of the traditional VRP, which regards only one type of propellant. The most cost-efficient routes for hybrid vehicles may differ substantially from corresponding routes for other types of vehicles. Our models consider a homogeneous fleet of plug-in hybrid vehicles, constrained by both load capacity limits and time windows for delivery. Recharging of the vehicle's battery is optional, and can be done only at special recharging sites (nodes). We consider the recharging times being either constant or dependent on the battery charge level at arrival at the recharging node.

The hybrid VRP has not yet been studied to a large degree. To the best of our knowledge, there is no successful implementation of a mathematical solution procedure for the hybrid VRP including separate customer and recharging nodes. The hybrid VRPs considered are modeled as mixed integer linear programs and solved using column generation, which separates each problem into a set covering master problem, and a shortest path subproblem.

Our tests show that the hybrid VRPs are time-consuming to solve exactly using conventional branch-and-cut methods. Our column generation approach combined with the dominance criteria reduces, however, the solution times considerably.

Keywords: hybrid vehicle routing problem, hybrid VRP, column generation, plug-in hybrid vehicles, recharging time, dominance criteria.

Acknowledgements

First of all I want to thank my supervisor, Peter Lindroth, for suggesting this engaging project and for all the advice and guidance he has provided. I also want to thank my examiner, Ann-Brith Strömberg, for her help through the project as well as for the courses on mathematical optimization she has provided during my two years at Chalmers.

I would further like to direct a thank you to my wife, Silvia, for all the support and joy she has given me throughout my years of studying. Finally I wish to thank my very supporting and loving family for always being there and for encouraging my curiosity for problem solving.

Jonathan Ruffieux, Gothenburg, November 2016

Contents

List of Figures

List of Tables

1
Introduction

Global warming, ozone layer depletion, water eutrophication, acid rain — the negative impacts man has had on the environment are many. Today, we are more aware of the consequences our lifestyles have on nature, and slowly our society has started to change to reduce these negative effects. In the United Nations' Climate Change Conference 2015, a global agreement on the reduction of climate change was created and finally signed by 174 countries in 2016 [1]. The main goal with the agreement is to restrict the global warming to less than two degrees Celsius. A lot of changes have to be done to reach these goals, not the least in the transportation sector. Still, in the European Union one-fifth of all CO_2 emissions are estimated to be caused by road transports, and according to the European commission the transport sector is the only major sector in which greenhouse gas emissions are still rising [2]. While vehicle manufacturers improve their products to make them more environmental friendly, an increase of emissions could likely be explained by a growing transportation sector. It is therefore likely that more efficient logistic planning, would have a great potential of reducing the hazardous effects the transportation sector has on the environment.

The idea of making transportation more efficient is not a new idea; the first vehicle routing problem (VRP) was introduced by Dantzig and Ramser already in 1959 [3] and is now a very well studied mathematical problem. This problem was formulated as a delivery problem, where a fleet of vehicles must be scheduled to satisfy the demand of some given costumers, while minimizing the total traveling distance. For a long time the main focus when studying different vehicle routing problems remained a pure profit optimizing one. But in later years, in tact with governmental regulations and appearance of new technologies, an additional focus on more direct environmental aspects of vehicle routing has been considered.

The way to go from studying environmental friendly routing problems, to implement these in large scale in real life may however be a long one. For a company it may not just be the big investment of the new fleet of vehicles, but they might be forced to adjust their business to a possible limited infrastructure for these vehicles in the areas where the company is active. This might especially be the case for electric vehicles, since they have a rather short reach per charge and are thus highly dependent on a dense net of recharging stations. As a possible easier option than changing the fleet fully to alternative fuel vehicles (such as electric, gas or ethanol), could be to consider the usage of hybrid vehicles, that is, vehicles that can run on more than one type of fuel. The possible advantages with this is that the new fleet, while still more environmental friendly, is less dependent on the new infrastructure than electric vehicles, for example. In this way the use of hybrid vehicles can serve

as a bridge between the use of old and new technology while making less of a change for companies.

Since 1959 many different variants of the VRP have been studied and solved by different techniques. VRPs with hybrid vehicles have not yet been widely researched, especially, not much has been done in terms of exact solution methods for these type of problems. In this thesis we therefore focus on the routing of plugin-hybrid vehicles (i.e. hybrid vehicles where the battery can be charged at recharging stations), for which we present two slightly different mixed integer linear vehicle routing models and also present a way to solve these using column generation combined with a dynamic programming algorithm. The subject was proposed and carried out as a collaboration between Chalmers University of Technology and Volvo Group Trucks Technology (GTT).

To be able to optimize their products it is of great use for Volvo AB to learn more about how their vehicles are used. With logistic planning for new types of vehicles, it may be possible to find driving patterns that are unique to these types of vehicles. This may give further insights of how the use of these new kinds of vehicles may differ form how other vehicles are used. With this report I hope to contribute by providing some more insight in the routing or hybrid vehicles. However, when modelling a real life problem mathematically, one must in general compromise between how well the model should describe reality, how easy it is to solve the problem and the ease of interpreting the results. In this report we therefore focus on adapting column generation methods for two rather idealized problems (one with the simpler assumptions of fixed recharging times, and the other where the recharging time depends on the battery level at arrival to the recharging station), future work in the area could suggestively be to apply this on more realistic data.

In this thesis we make the following restrictions:

1. The fuel and the electricity usage are assumed to be directly proportional to the distance travelled with the respective propellant.
2. When a recharge is undertaken it must be continued until the battery is full.
3. We also assume that one freely can choose when to drive on what propellant.

The outline of this thesis is as follows. First, in Chapter 2, a literature study is conducted, which gives an overview of the literature of VRPs that puts environmental questions in focus. Then we describe the two hybrid vehicle models that will be treated in this thesis, along with a mathematical formulation of each of them. Chapter 4 will present some optimization methods that will be used to solve the problems and Chapter 5 decomposes the two hybrid VRPs into a column generation form in terms of master- and subproblems. In Chapter 6 the implementation of the column generation method for the hybrid VRPs are presented. The largest part of this chapter is devoted to the solution method of the subproblems, which is an dynamic programming algorithm. For this dynamic programming algorithm we present problem specific dominance criteria, which are stronger than the dominance criteria generally used when solving VRP. In Chapter 7 the tests and results are presented and in Chapter 8 the conclusions and outlooks are presented.

2

Literature Study

Since around the beginning of the 21st century a new focus for VRPs has been researched, the so called *green VRPs*. The green VRPs are simply VRPs that takes environmental aspects, like greenhouse gas emissions, into account. This chapter presents an overview of the study of green VRPs. Section 2.1 gives a brief overview of areas in which green VRPs that have been studied, while Section 2.2 specifically treats the written work on routing of hybrid vehicles.

2.1 A Brief Overview of green VRP Literature

Today, there is an increased focus on the environment and a pressure to decrease the amount of emissions of greenhouse gases. This can be seen in an increase of publications of VRP literature with focus on different environmental aspects, starting fifteen years back ([4] contains a review of publications until 2013 aiming at the reduction of fuel consumption).

In this text existing surveys on the green VRPs are summarized and some new interesting literature on the green VRPs is added.

The survey by Choy et al. 2014 [5] describes green VRP (denoted in this text as G-VRP) as follows,

> "G-VRPs are characterized by the objective of harmonizing the environmental and economic costs by implementing effective routes to meet the environmental concerns and financial indexes."

and divides the green VRP into three groups:

- "Green VRP" (G-VRP): Deals with the optimization of energy consumption of transportation
- Pollution Routing Problems (PRP): VRP that aims to find routes with less pollution
- VRP in Reverse Logistics: Focus on the distribution aspects of reverse logistics[1]

In Bektaş et al. 2016 [6], however, the green VRP is described as:

> "Green vehicle routing is a branch of green logistics which refers to vehicle routing problems where externalities of using vehicles, such as carbon dioxide-equivalents emissions, are explicitly taken into account so that they are reduced through better planning."

[1] The logistics that is concerned with collection and recycling of waste products.

2.1.1 Limitations

Neither in [6] nor in the earlier literature survey by Bektas, Demir and Laporte [4] is reverse logistics regarded as a part of green VRP. The decision not to include these problems in the definition of green VRP seems reasonable as these problems are not mainly concerned with environmental-friendly vehicle routing. The "green part" in VRP in Reverse Logistics is mainly about waste and end-of-date goods collection for recycling and not environmental friendly routing. I have thus decided to review the literature as follows. I will present literature from and related to the groups Green VRP ands PRP presented in [5], however I will not sort them into separate groups as the line between them gets very vague. The literature concerning reverse logistics is left out.

2.1.2 Green VRP literature

Energy minimization A rather straightforward way to approach the influence on the environment in a VRP is to directly minimize energy consumption.

This was done by Kara et al. [7] in what they call the Energy Minimizing Vehicle Routing Problem (EMVRP), modelled as an integer linear program. They formulated a capacitated VRP (CVRP), in which the nodes have demands that must be met and the vehicles have capacity limits. The objective is to minimize the total energy consumed (total work done by the vehicles) instead of the total distance travelled or total cost of the routes. The work done by the vehicle is calculated as the work done by overcoming the vehicle's friction forces and was thus assumed only to depend on the weight of the vehicle and some friction constant. For a set of test instances, solutions to the EMVRP were compared with solutions to corresponding distance minimizing VRP; the latter model resulted in up to 13% shorter paths. For a set of test instances, solutions to the EMVRP were compared with solutions to corresponding distance minimizing VRP; the latter model resulted in up to 13% shorter paths. Obviously,the most environmental friendly route is not necessarily the shortest one, which shows the need for new objectives to study the environmental aspects of vehicle routing.

Fukasawa et al. (2016) [8] present two alternative mixed integer linear programming formulations of the EMVRP presented in [7]. They also propose two exact solution methods; one branch-and-cut algorithm (see [9, Ch. 12]), and one branch-cut-and-price algorithm (see [9, Ch. 13]) for the EMVRP, the latter showing especially good performance.

Fuel/Emission minimization A more commonly used approach to reduce the environmental impact is to minimize the fuel consumption or the emissions directly. However, both the fuel consumption and the total emissions of a vehicle are very complex to model. In [10] the authors put several different fuel consumption models from the literature to test. Six different models describing fuel consumption under constant speed were tested w.r.t. how well some collected data was predicted. The data used varied in both mean speed and in total weight of the vehicle. The results showed that none of the models could describe the collected data for all instances.

Four of the models always overestimated the fuel usage while the two others underestimated it. This suggests that one ought to be careful when interpreting results from literature where fuel consumption estimation plays a great roll. The results also show that it is hard to compare models in terms of "success of reducing emissions" if their fuel consumption models differ. However, in [10] the behavior of the models was also analyzed separately. The conclusion was that all of them follow the general properties of fuel consumption: increased fuel consumption per km at *low speeds* due to inefficient energy usage, at *high speeds* due to increased air drag, and for *heavier vehicles* due to increased friction.

The article [4] also reviews fuel consumption models. It also lists factors that have been shown to influence the fuel consumption. A short version of the list of factors is (in the order from most to least commonly used in the literature): *Speed, Payload, Congestion, Fleet Size and Mix, Gradient, Driver Profile.*

Bektas and Laporte [11] present the so-called Pollution Routing Problem (PRP) as an extension of the VRP with time windows (VRPTW), which takes a broader view on the costs involved.[2]. The PRP is modelled as a non-linear mixed-integer optimization problem which is linearized and solved for small instances with objectives that include fuel, emission, and driver costs, using speed and load as decision variables. Demir et al. [12] also develop a two-stage adaptive large-neighborhood heuristic to solve this PRP for large instances. They also introduced a new set of test instances for PRP and on which the heuristic is tested. Koç et al. [13] further extends the PRP by considering a heterogeneous vehicle fleet, which they solve using a metaheuristic.

Xiao et al. [14] present the Fuel Consumption Rate CVRP (FCVRP) where the total fuel consumption is minimized. As in [7], the only factors affecting the cost (in this case in terms of fuel consumption) of a route are the distance traveled and the vehicle weight. The fuel consumption is calculated using a Fuel-Consumption-Rate that depends on the load, and is modeled using linear regression on statistical data. The model is solved for some test instances for CVRPs with Fuel-Consumption-Rate and the results show that the FCVRP model can reduce fuel consumption by 5% on average compared to the regular CVRP model.

Kuo [15] presents a time dependent VRP (TDVRP) that aims to minimize fuel consumption. In this model the vehicle speed is time dependent and thus the traveling time between nodes varies over time. The model fulfills a non-passing property, which means that leaving a node earlier must result in also reaching the next node earlier. The fuel consumption is here modeled to depend both on the weight of the vehicle and its travel speed. The problem is solved using simulated annealing and the results suggest a 24.6% and 22.7 % reduction in fuel consumption compared with the method based on minimizing travel time and distances, respectively.

Demir et al. [16] presented a bi-objective version of the PRP which aims to minimize the total travel time and the fuel consumption. An adaptive large-neighborhood search algorithm combined with a speed optimizing procedure is used to solve this problem. Test results show that energy savings can be done at the cost of travel time, showing that 9.7% increase in driving time can decrease the energy require-

[2]A VRPTW is a VRP in which the customers specify time intervals during which they should be visited.

ment by 27%.

Another view of time dependent speeds and emissions is to consider traffic congestions. Maden et al. [17] present a CVRP with the objective to minimize the total travel time under influence of congestion. The case study included showed about 7% reduction in CO_2 emissions as compared to a model that assumed constant speed.

Franceschetti et al. [18] contribute with a model that minimizes fuel consumption while also taking congestion into consideration. The model is an extension of the PRP presented in [11] and is presented as a time dependent integer linear program, using the fuel consumption model presented in [11]. The congestion part is modeled in a very simple way; two time periods are considered, the first with congestion and the second without. Speed and departure times for customers are decision variables in the model.

A later article in the same direction of study is presented by Xiao and Konak [19], who formulate a time-dependent heterogeneous G-VRP. The objective is to minimize CO_2 emissions and a weighted tardiness penalty for delivery after due date. The emission rate is not modeled to depend on payload, but they are assumed to be time and vehicle dependent. The authors also present a meta-heuristic combining a genetic algorithm for the routing part of the problem with a dynamic programming algorithm for the scheduling part, seemingly with good results.

Yet another publication that models congestion is [20]. The speed is considered as a decision variable and the objective is to minimize greenhouse gas emissions. The authors present a column generation based tabu search algorithm to solve the problem and the results show a reduction in emissions by 3% as compared to when the total travel time is minimized.

Alternative-Fuel Powered VRP A relatively new field that has become popular to study is alternative-fuel VRP (AF-VRP). These problems model, for example, electrically or natural gas driven vehicles, which can help to reduce the negative effects freight transports has on nature. There has been a lot of studies concerning different kinds of vehicles in VRPs through the years, especially in the heterogeneous VRPs, where the vehicle capacity is one of the parameters that has been varied most. The usage of alternative fuel vehicles differs from conventional fuel vehicles in most of these studies in the aspect that they have a limited driving range or that refueling stations are in general scarce. There is thus a big difference between these problems and the traditional VRPs as AF-vehicles can get stranded if the routes are not short enough or if refueling is not planned ahead. This means that the AF-VRP in general can not be modeled as a traditional VRP or be solved by the same solution methods.

Erdogan and Miller-Hooks [21] present a VRP, in which there is a recharging/refueling option for the AF-vehicles to be be able to use them for longer routes. The objective is to minimize the total travel distance and the problem is modelled as a mixed integer linear program. The fuel consumption is modeled to depend only on traveled distance and vehicle weight, and a fixed recharging/refueling time is assumed. Two different heuristics are suggested to solve the problem: a Modified Clarke-and-Wright Savings Heuristic (see [22]) and a Density-Based Clustering Algorithm.

Schneider et al. [23] suggest a mixed integer linear program for electrical vehicles

with time windows which considers non-fixed charging times at recharging stations. Here, the time it takes to charge the battery is dependent on how much charge there is left when reaching the recharging station. The recharging speed is for the simplicity of the model assumed to be constant, even though the last 10–20% of the charge quantity process takes longer time (Marra et al. [24]). As solution method they present a hybrid heuristic combining a variable neighborhood search algorithm with a tabu search heuristic.

Further AF-VRP literature includes among others [25], in which the battery consumption is modeled more thoroughly (so it depends on vehicle speed, road slopes, and vehicle weight), [26] and [27], in which bi-fuel vehicles are used, and [28], which considers the possibility to recharge electric vehicles at all customer nodes.

2.2 Literature on VRP with plug-in hybrid vehicles

In this section some more details on literature concerning VRP with plug-in hybrid vehicles is given. To the best of our knowledge only three written works exist in this subject. These are a MSc thesis by Abdallah [29], a paper by Lebeau et al. [30], and an unpublished paper by Mancini [26]. A summary of these works is given here along with some short notes on how they relate to this thesis.

The VRP considered in [29] assumes that the vehicle batteries can be recharged at any customer node. The recharging can take place either before or after a customer is served and partial recharging of the batteries is allowed (i.e. the recharging time is a decision variable). The objective is to minimize the cost arising from travelling on conventional fuel and no cost is considered for driving on electricity. Both a tabu-heuristic and a commercial solver are used to solve different Lagrange relaxed versions of the problem.

The paper [30] takes another modelling approach compared to [29]. Here, recharging of vehicles can only be undertaken at the depot, and the recharging time depends linearly on the battery level. The objective is to minimize the total cost, which is the sum of vehicle cost, propellant cost, and cost of drivers. The propellant cost is considered to be a fixed price per kilometer, hence it does not make any difference between the cost of running on conventional fuel and running on electricity. A savings heuristic similar to the one presented in [22] is used to solve the problem.

In [26], recharging of vehicles may be undertaken only at special recharging stations, similar to the AF-VRPs presented in [21, 23]. A difference to those works is, however, that there is nothing that forces a visit to a recharging node, since a plug-in hybrid vehicle can simply switch propellant when needed to. The recharging time is here assumed to be constant and driving on conventional fuel corresponds to a cost in the objective function. The solution method suggested in [26] is a Large Neighborhood Search heuristic.

Both [30] and [26] have, however, some inconsistency between the models described in the texts and their corresponding mathematically formalized models. In [30] nothing seems to promote the usage of electricity, which results in optimal solutions where electricity is never used. In [26] only non-positive battery levels are

allowed, by resetting the battery level to zero in recharging nodes; this results in a strange objective penalty cost, which in the objective function that depends on the battery level and the distance to the closest recharging node.

A paper that closely relates to hybrid VRPs is [31] by Arslan et al., presenting a shortest path problem (SPP) with plug-in hybrid vehicles—containing both recharging nodes and battery switching nodes. In addition, in [32] by Arslan et al. the problem of placing recharging nodes for the SPP with plug-in hybrid vehicles is presented. Both these problems are solved using heuristic solution methods.

The hybrid VRPs in this thesis will, similarly to [26], consider special battery recharging nodes. We present two problem versions, one with fixed recharging times and the other with recharging times that directly depend on the battery level at arrival to the recharging node. In both problems the objective is to minimize the total cost of propellant used (both electricity and conventional fuel) while serving the customers. The solution method used is a column generation method, in which the subproblems—elementary shortest path problems with resource constraints—are solved with a dynamic programming algorithm. For comparison, the original problem formulations (being network flow models) are also solved directly with a commercial solver.

3
Problem formulation

In this chapter the two hybrid vehicle routing problems that are considered in this thesis are formalized, along with their flow representations. Later, in Chapter 5 set covering formulations of the problems will be presented (in terms of master and subproblems resulting from a Dantzig-Wolfe decomposition).

3.1 Two hybrid vehicle routing problems

The hybrid vehicle routing problems treated in this report can be regarded as relaxed electrical vehicle routing problem, such that driving with an empty battery is possible but corresponds to another (higher) cost per mileage. Due to these similarities almost the same problem setup is regarded in our case as for the electrical VRPs described in [23].

In terms of a graph $\mathcal{G}$ our problem contains three different types of nodes: First there is the depot, node 0, at which all vehicles start and end their routes; second the set of recharging nodes, $\mathcal{V}_{\text{rec}} = \{1, \ldots, m\}$ containing the nodes in which the vehicles are able to recharge their batteries; and finally the set of customer nodes, $\mathcal{V}_{\text{cus}} = \{m+1, \ldots, n\}$ which all must be visited to get their demands met. Each customer node has a time window related to it which specify between what times the node must be visited, as well as a certain demand that must be satisfied by the visiting vehicle. The depot and the recharging nodes have no demands to be met, however they do also have time windows; for the depot this time window is the period of time in which all routes must be covered, and for each recharging node it says if a battery recharge can start.

The way the hybrid VRP differs from the electrical VRP is only in the kind of vehicles used. While the electrical VRP considers vehicles that only can drive on battery, we consider plug-in hybrid vehicles which can run on either battery or some conventional fuel like diesel. We assume that the driver can choose freely what propellant to use at what time; if the battery is empty, however, the only choice is to run on fuel. Figure 3.1 shows an example of a hybrid VRP where the routes shown may be partly covered by the use of electricity and partly by conventional fuel, and Figure 3.2 shows the same example for electric vehicles.

The objective of our hybrid VRPs is to minimize the total cost while assuming that the cost of running on electricity, in terms of cost per kilometer, is cheaper than to run on conventional fuel.[1] However, as a downside the maximum driving

[1]This is the case today in Sweden and has been for a while due to the fact that conventional fuels are heavily taxed.

distance of one full battery charge is quite limited at the same time as a recharging takes long time, and the recharging stations remain rather scarce. This may make it beneficial to use a mixture of both the cheaper option (driving on electricity) and the more expensive, but less time consuming, option of using conventional fuel.

The vehicle fleet we consider is homogeneous, meaning that only one kind of vehicle is present in the fleet. Each vehicle starts at the depot with a fully charged battery, with a fuel level regarded as unlimited[2], and cargo loaded to its maximum capacity. Finally, we allow each vehicle to visit each recharging node at most once, and that when a recharge is undertaken it may not be stopped before the battery capacity is reached.

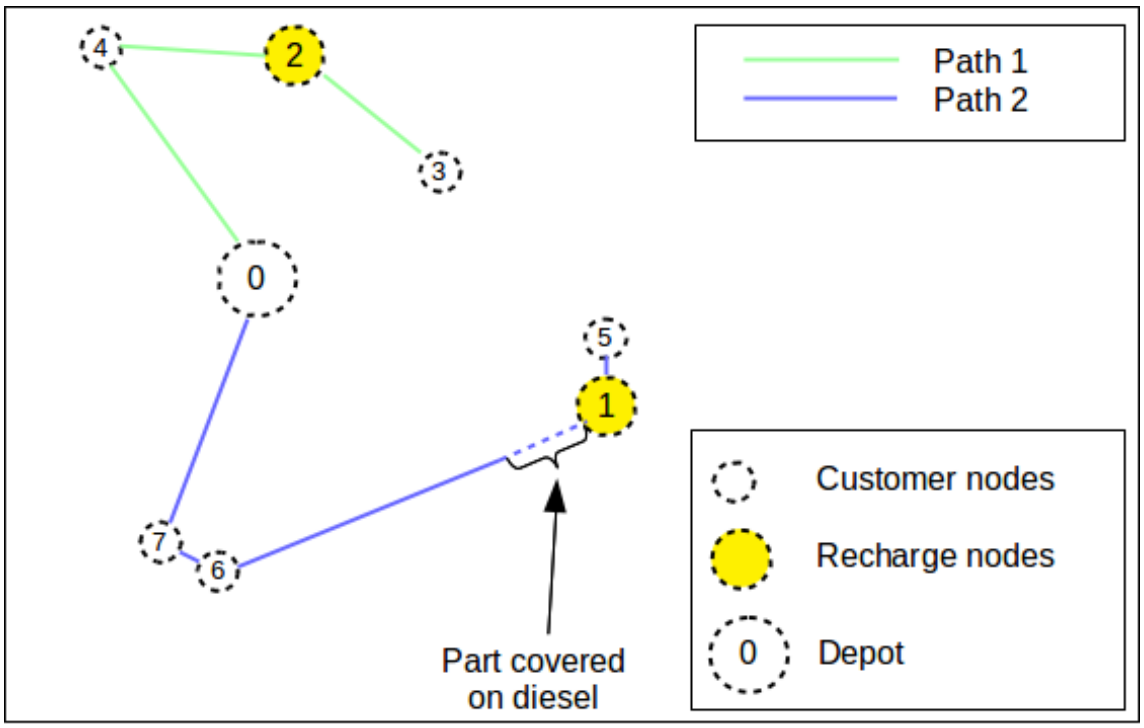

Figure 3.1: Example of problem setup for a hybrid VRP with different nodes. The last line for each vehicle route that connects it to the depot is admitted to show in what direction the vehicle moves. The dashed line in this illustration shows the part covered on diesel instead of battery.

[2]The reason why we regard the fuel level for a vehicle to be unlimited is that, a refueling takes much shorter time than a recharge. Further, one usually do not have to take long detours to find a refueling station, as is assumed often to be the case for recharging stations.

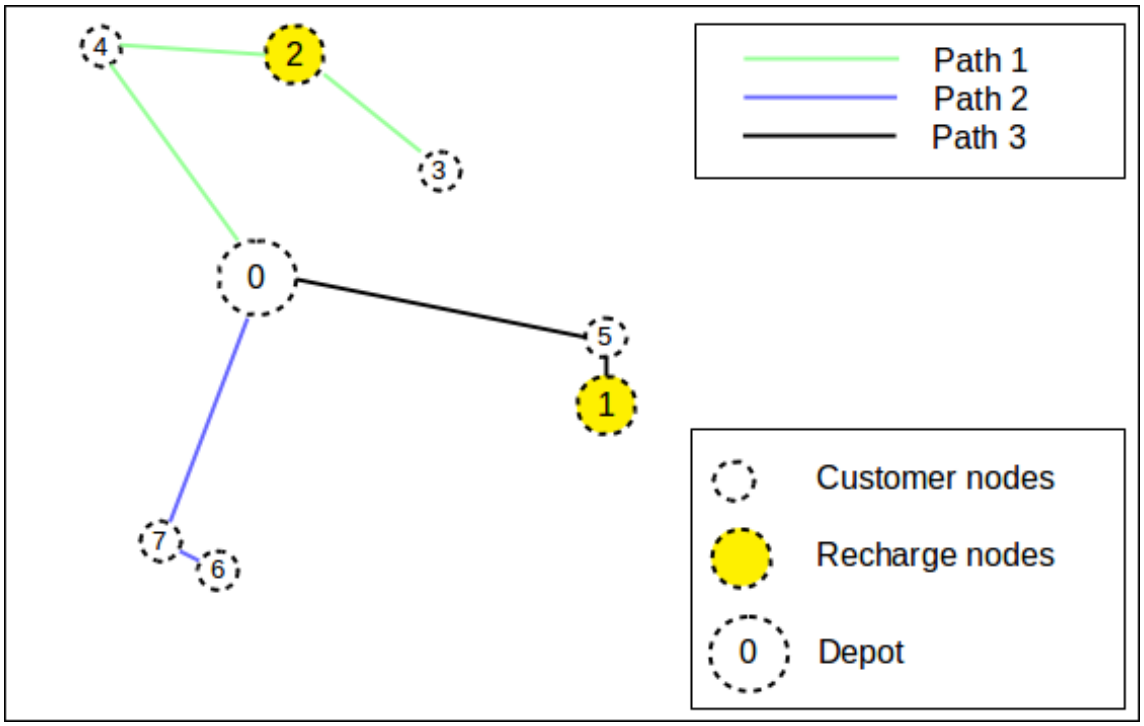

Figure 3.2: Example of a solution the problem in Figure 3.1 for electric vehicles (with the same battery capacity as the hybrid vehicles considered in Figure 3.1). As no part can be covered with diesel a third vehicle is needed to visit all nodes.

In this thesis we look at two different problems. Both describing the routing of hybrid vehicles, but which models the recharging time in different complexity. The simpler problem, Problem A, in which the recharging time is set to the time for a full recharge, and the more complex one, Problem B, the recharging time depends on the battery level upon arrival at the recharging node. The modelling of recharging time is important as a full recharge of a vehicle is very time consuming. Problem B is more accurate in the description of recharging times than Problem A, but as a consequence Problem B is also harder to solve. How the recharging time is modelled really makes a difference, and because of this we will look at both of these problems. The two problems are modelled below using flow formulations.

3.2 Flow formulations of the hybrid VRPs

In this section the two hybrid VRP problems described in the previous section are presented mathematically as vehicle folow formulations. In Table 3.1 sets, parameters, and variables needed to formulate our problems are given.

Table 3.1: Definition of sets, parameters and variables used for formulation of Problems A and B

Notation	Description
Sets	
$\mathcal{V} = \{0\} \cup \mathcal{V}_{\text{rec}} \cup \mathcal{V}_{\text{cus}}$	The set of all nodes in the graph
$\mathcal{K} = \{1, \ldots, F_{\max}\}$	The set of all vehicles
$\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$	The set of all arcs in the graph
Parameters	
$F_{\max}$	Size of vehicle fleet (number of vehicles)
$T_{\max}$	Latest time be return to the depot [h]
$U_{\max}$	Cargo storage capacity in each vehicle [kg]
$Q_{\max}$	Battery capacity [kWh]
$c^{\text{fu}}/c^{\text{el}}$	Cost of fuel/electricity [€/litre]/[€/kWh]
$r^{\text{fu}}/r^{\text{el}}$	Consumption rate of fuel/electricity [litre/km]/[kWh/km]
t_{ij}	Traveling time over arc $(i,j) \in \mathcal{A}$ [h]
p_i	Service time at node $i \in \mathcal{V}$ [h]
q_i	Demand of cargo in node $i \in \mathcal{V}$ [kg]
d_{ij}	Length of arc $(i,j) \in \mathcal{A}$ [km]
e_i	Earliest time at which the service in node $i \in \mathcal{V}$ can start [h]
l_i	Latest time at which the service in node $i \in \mathcal{V}$ can start [h]
M	A large enough number [h]
Variables	
x_{ij}^k	$= 1$ if arc $(i,j) \in \mathcal{A}$ is used by vehicle $k \in \mathcal{K}$, $= 0$ otherwise
τ_i^k	Arrival time at node $i \in V \setminus \{0\}$ for vehicle $k \in \mathcal{K}$ [h]
u_i^k	Amount of cargo in vehicle $k \in \mathcal{K}$ at arrival in node $i \in \mathcal{V}$ [kg]
y_i^k	Battery level for vehicle $k \in \mathcal{K}$ upon arrival in node $i \in \mathcal{V}$ [kWh]
$z_{ij}^{\text{Fu},k}/z_{ij}^{\text{El},k}$	Amount of fuel/electricity used on arc $(i,j) \in \mathcal{A}$ by vehicle $k \in \mathcal{K}$ [litre]/[kWh]

Here it is enough if parameter $M = 2T_{\max}$. This is the case as we will only solve the problems for test instances where the travelling time between two nodes is less than $2T_{\max} - \max_{i \in \mathcal{V}}\{p_i\}$. Further, the service time, p_i, for recharging nodes we define as the time it takes for a full recharge.

Problem A: The hybrid VRP with fixed recharging time
Given the notations in Table 3.1, problem A is defined as to

$$\underset{x,\tau,u,y,z}{\text{minimize}} \quad \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} \left(c^{\text{Fu}} z_{ij}^{\text{Fu},k} + c^{\text{El}} z_{ij}^{\text{El},k} \right), \tag{A1}$$

subject to

$$\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij}^k = 1, \qquad i \in \mathcal{V}_{\text{cus}}, \tag{A2}$$

$$\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij}^k \leq 1, \qquad i \in \mathcal{V}, k \in \mathcal{K}, \tag{A3}$$

$$\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij}^k = \sum_{j\in\mathcal{V}:(j,i)\in\mathcal{A}} x_{ji}^k, \qquad i \in \mathcal{V}, k \in \mathcal{K}, \tag{A4}$$

$$\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{V}_{\text{rec}}\cup\mathcal{V}_{\text{cus}}} x_{0j}^k \leq F_{\max}, \tag{A5}$$

$$e_i \leq \tau_i^k \leq l_i, \qquad i \in \mathcal{V}, k \in \mathcal{K}, \tag{A6}$$

$$T_{\max} - (t_{i0} + p_i) \geq \tau_i^k, \qquad i \in \mathcal{V} \setminus \{0\}, k \in \mathcal{K}, \tag{A7}$$

$$\tau_i^k + (t_{ij} + p_i) - M(1 - x_{ij}^k) \leq \tau_j^k, \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}, k \in \mathcal{K}, \tag{A8}$$

$$u_0^k \leq U_{\max}, \qquad k \in \mathcal{K}, \tag{A9}$$

$$u_i^k - q_i x_{ij}^k + U_{\max}(1 - x_{ij}^k) \geq u_j^k, \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}, k \in \mathcal{K}, \tag{A10}$$

$$z_{ij}^{\text{Fu},k}/r^{\text{Fu}} + z_{ij}^{\text{El},k}/r^{\text{El}} = x_{ij}^k d_{ij}, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}, \tag{A11}$$

$$y_i^k - z_{ij}^{\text{El},k} + Q_{\max}(1 - x_{ij}^k) \geq y_j^k, \qquad i \in \mathcal{V}_{\text{cus}}, j \in \mathcal{V}, k \in \mathcal{K}, \tag{A12}$$

$$Q_{\max} - z_{ij}^{\text{El},k} + Q_{\max}(1 - x_{ij}^k) \geq y_j^k, \qquad i \in \mathcal{V} \setminus \mathcal{V}_{\text{cus}}, j \in \mathcal{V}, k \in \mathcal{K}, \tag{A13}$$

$$x_{ij}^k \in \{0, 1\}, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}. \tag{A14}$$

$$z_{ij}^{\text{El},k}, z_{ij}^{\text{Fu},k} \geq 0, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}, \tag{A15}$$

$$\tau_i^k, u_i^k, y_i^k \geq 0, \qquad i \in \mathcal{V}, k \in \mathcal{K}. \tag{A16}$$

The objective (A1) of the model is to minimize the total driving cost for the fleet of vehicles. Constraints (A2)–(A4) are the node balance constraints, where constraint (A2) ensure that each customer node is visited once, (A3) that each vehicle may visit any node at most once, and (A4) that a vehicle that arrives at a node must also leave the same node. The number of vehicles used is limited by (A5) and the arrival times are given by (A6)–(A8); where (A6) and (A7) assure that every node is visited within its time window and (A8) defines the order of arrival times at nodes and by this also prohibits subtours. The demand at nodes is met without overriding the cargo capacity of the vehicle, according to (A9) and (A10), and the battery charge level at arrival at a node and the usage of electricity between two nodes is regulated by (A11)–(A13). Constraint (A11) says that enough propellant must be used to cover the distances travelled, while (A12) and (A13) gives the battery level for each vehicle and node. Finally, (A14) says that each arc is either travelled on or not by vehicle k, (A15) that we can not use a negative amount of propellant, and (A16)

says that the battery level and the amount of cargo is never negative, and that the arrival time at any node is after time 0h.

Problem B: The hybrid VRP with the recharging time dependent on the battery level

To model the recharging time as dependent of the battery charge level at arrival of a node, we introduce the variable s_i^k for the recharging time for vehicle k in node $i \in \mathcal{V}_{\text{rec}}$, then let the parameter g denote the recharging rate (h/kWh). Problem B can then be expressed as Problem A where the time constraints (A7)–(A8), for each $k \in \mathcal{K}$, are replaced by the constraints

$$\tau_i^k \leq T_{\max} - (t_{i0} + p_i), \qquad i \in \mathcal{V}_{\text{cus}}, \tag{B1}$$
$$\tau_j^k \geq \tau_i^k + (t_{ij} + p_i) - M(1 - x_{ij}^k), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V} \setminus \mathcal{V}_{\text{rec}}, \tag{B2}$$
$$\tau_i^k \leq T_{\max} - (t_{i0} + s_i^k), \qquad i \in \mathcal{V}_{\text{rec}}, \tag{B3}$$
$$\tau_j^k \geq \tau_i^k + s_i^k + t_{ij} - M(1 - x_{ij}^k), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}_{\text{rec}}, \tag{B4}$$
$$s_i^k \geq g(Q_{\max} - y_i^k), \qquad i \in \mathcal{V}_{\text{rec}}, \tag{B5}$$
$$s_i^k, \tau_i^k \geq 0, \qquad i \in \mathcal{V}. \tag{B6}$$

The constraints (B1) and (B2) fill the same functionality as (A7) and (A8) in the flow formulation of Problem A, while in constraints (B3) and (B4) the service time is replaced by a recharging time variable. Finally, the constraint (B5) makes the recharging time long enough to fully recharge the battery, and (B6) says that both the arrival time at all nodes and all recharging times are non-negative.

4
Methodology

In this section we describe some theory that can be used to solve different types of linear optimization problems. First we present the *simplex method*, which can be used to solve linear problems with continuous variables. Then we look at the *branch-and-bound* and *branch-and-cut* methods, which can be used to solve integer linear problems (ILP). In the last part of this chapter we present the basics of *column generation*, which is a decomposition method that can be used to tackle very large linear programs. The column generation method is primarily aimed for solving linear optimization problems with continuous variables. But it is also possible to apply it to mixed integer optimization problems (MILP), like Problem A and Problem B, defined in Chapter 3. In this case, however, there will be no guarantee that an optimal solution is found.[1]

4.1 Simplex method

Linear programs (LP) are often solved by some variant of the simplex method. We present the basics of the simplex method, based on the description in [34].

Consider the linear program to

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & z = c^{\mathrm{T}}x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{4.1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$. This problem is stated in its standard form.[2] Assume that rank $A = m$; then, if $m < n$, a solution to $Ax = b, x \geq 0$ (if one exists) can be found for which at most m variables are non-zero; we denote these m variables as x_B (these are called *basic* variables). Further let x_N denote the $n - m$ *non-basic* variables (all taking the value zero). This kind of solution is called a *basic feasible solution* (BFS), while more generally such a solution to the system $Ax = b$, without the sign restriction of x, is called a *basic solution* (BS) to (4.1). For any BS is possible to partition the matrix A into the matrices N and B, where we let the matrix $N \in \mathbb{R}^{m \times (n-m)}$ consist of the columns of A that corresponds to the non-basic

[1] Given enough time and memory it is possible to guarantee an optimal solution even for an integer optimization problem if the column generation is incorporated with the branch-and-bound method, leading to what is called the *branch-and-price* method [33]. This is however not described in this report.

[2] Any linear program can be rewritten into this form by the introduction of slack variables.

variables x_N, and the matrix $B \in \mathbb{R}^{m \times m}$ (being of full rank) consists of the columns corresponding to the basic variables x_B. Given this partition of the matrix A, the program (4.1) can be rewritten as to

$$\begin{aligned} \text{minimize} \quad & z := c_B^{\mathrm{T}} x_B + c_N^{\mathrm{T}} x_N, \\ s.t. \quad & Bx_B + Nx_N = b, \\ & x_B, x_N \geq 0. \end{aligned} \tag{4.2}$$

Given a BFS to (4.2), the next step in the simplex method is to find a non-basic variable that will improve the objective value if introduced among the basic variables. If no such variable can be found then the present BFS is an optimal solution to (4.2). To find out if it is possible to improve the solution, and in that case also what variable to include in the basic variables, we take a closer look at the objective function. Let us therefore start by writing the objective function (z) in terms of non-basic variables only: First, x_B can be expressed in terms of x_N, since $Bx_B + Nx_N = b \Leftrightarrow x_B = B^{-1}(b - Nx_N)$, and inserting this in the expression for z we get $z = c_B^{\mathrm{T}}(B^{-1}(b - Nx_N)) + c_N x_N = c_B^{\mathrm{T}} B^{-1} b + (c_N^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} N) x_N$.

The new expression for our objective function shows that a non-basic variable, x_j, would improve the objective value when its value is increased (i.e., if it was turned into a basic variable) only if $(c_N^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} N)_j < 0$. The expression $\tilde{c}_j := (c_N^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} N)_j$ is called the *reduced cost* of the variable x_j. For example, one could iteratively pick the non-basic variable with the lowest reduced cost to enter the set of basic variables until an optimum (or unboundedness) is found. Each time a non-basic variable is added to the set of basic variables an old basic variable must leave the basis. When the value of $x_j \in x_N$ is increased it affects the values of the variables in x_B as $x_B = B^{-1}(b - Nx_N)$. Given that $(B^{-1}N_j)_i > 0$ for some $x_i \in x_B$, the variable to leave x_B is the one which value $(B^{-1}(b - N_j))_i$ reaches 0 first when increasing the value of x_j (i.e., $x_i \in x_B$ with the smallest value of $(B^{-1}b)_i / (B^{-1}N_j)_i$ while $(B^{-1}N_j)_i > 0$).

4.2 Branch-and-bound and Branch-and-cut

ILPs are in general much harder to solve than LPs; many of these problems are actually NP hard (see, e.g., [35]) while there exists polynomial time methods for solving LP ([36]). A method that can be used to solve ILPs and MILPs is the *branch-and-bound* method. The idea behind this method is to exploit the fact that an LP is relatively easy to solve. The branch-and-bound method therefore iteratively divides the problem into subproblems which are solved after relaxing their integrality constraints. Another popular method is the *branch-and-cut* method, which is a generalization of the branch-and-bound method, where extra constraints can be added to enhance the convergence to an integer solution. In this section we first look at the branch-and-bound method after which we look at an additional cut that can be used for the branch-and-cut method. The theory presented in this section is based on [9, Ch. 11–12].

4.2.1 Branch-and-bound

Consider the following ILP (S) and its relaxation to an LP (S_{LP}):

$$\begin{aligned} \underset{y}{\text{minimize}}\ & z := c^{\mathrm{T}}y, \\ \text{s.t. } & A^{\mathrm{T}}y = b, \qquad (S) \\ & y \in \mathbb{Z}_{\geq 0}, \end{aligned} \qquad \begin{aligned} \underset{y}{\text{minimize}}\ & z_{LP} := c^{\mathrm{T}}y, \\ \text{s.t. } & A^{\mathrm{T}}y = b, \qquad (S_{\text{LP}}) \\ & y \geq 0. \end{aligned}$$

For the problems (S) and (S_{LP}) we define the following notations:

$$\begin{array}{lcl} y^* & = & \text{an optimal solution in } S, \\ z^* & = & \text{optimum objective value for } S, \\ \overline{z} & = & \text{lowest upper bound on } z^* \text{ found}, \\ \underline{z} & = & \text{highest lower bound of } z^* \text{ found}, \\ y_{\text{LP}} & = & \text{an optimal solution to } S_{\text{LP}}, \\ z_{\text{LP}} & = & \text{the optimal objective value for } S_{\text{LP}}, \\ \Theta & = & \{y \in \mathbb{Z}_{\geq 0} \mid A^{\mathrm{T}}y = b\}, \text{ the solution space for } S \\ \Theta_{\text{LP}} & = & \text{the solution space for } S_{\text{LP}}. \end{array}$$

In the solution process, S is divided into subproblems, such that the combined solution space of all subproblems is the same the solution space for S, while the combined solution space for all integer relaxed subproblems is a subset of the solution space to S_{LP}. The LP relaxed subproblems are then solved to improve the knowledge about optimal solutions of S. We let S^k denote the kth subproblem of S and S^k_{LP} its LP relaxation. The branching of the problem S into subproblems, which in turn are branched in to new subproblems yields a tree structure of problems, where each problem is commonly referred to as a *node*. Further some more notations

$$\begin{array}{lcl} z^k & = & \text{optimum objective value for } S^k, \\ \overline{z}^k & = & \text{lowest upper bound found on } z^k, \\ \underline{z}^k & = & \text{highest lower found on } z^k, \\ y^k_{\text{LP}} & = & \text{an optimal solution to } S^k_{\text{LP}}. \\ z^k_{\text{LP}} & = & \text{the optimal objective value for } S^k_{\text{LP}}, \\ \Theta^k & = & \text{the solution space for } S^k, \\ \Theta^k_{\text{LP}} & = & \text{the solution space for } S^k_{\text{LP}}. \end{array}$$

With these notations the Branch-and-bound procedure for solving S can be described as follows.

Algorithm 1 Branch-and-bound

1. **Initialize bounds:** Let the highest lower bound of z^* be $\underline{z} := -\infty$ and the lowest upper bound be $\overline{z} := \infty$.

2. **Solve S_{LP} and update bounds:** Solve S_{LP}. If the solution is integer, terminate as this solution is also optimal to S. If S_{LP} is infeasible, terminate as S is also infeasible. Otherwise, update the lower bound $\underline{z} := z_{\text{LP}}$ and go to step 3.

3. **Branch S:** Branch the problem S into subproblems (S^k) in such a way that $y^* \in \bigcup_k \Theta^k$ and $y_{\text{LP}} \notin \bigcup_k \Theta^k_{\text{LP}}$. This can be done in many different ways, but a common way is to branch it on a single variable, y_j, which does not take an integer value in y_{LP}. Let $y_j = y_j^0$ be this non–integer value, then the two new problems that this node branches into each contain an additional constraint; one with the constraint $y_j \geq \lfloor y_j^0 \rfloor$, and the other with the constraint $y_j \geq \lceil y_j^0 \rceil$.

4. **Choose an untreated node:** Pick a node S^k that has not yet been treated.

5. **Solve S^k_{LP} and update bounds:** Solve S^k_{LP}. If y^k_{LP} is an integer solution and $z^k_{\text{LP}} < \overline{z}$ then update the upper bound to $\overline{z} := z^k_{\text{LP}}$.

6. **Prune node k if possible:** Prune the node k if possible, i.e., regard it as fully treated with no need to branch the problem further. A node S^k can be pruned for one of three different reasons:
   (a) **Prune due to infeasibility.** If S^k is infeasible it cannot contain an optimal solution to S and can thus be disregarded.
   (b) **Prune because of integer solution.** If solving S^k_{LP} results in an integer solution there is no need to continue branching node k as this could not result in a better integer solution.
   (c) **Prune because of worse bounds.** If the lower bound from S^k is worse (higher) than the best known upper bound for the problem S, an optimal solution to S can not exist in node k.

7. **Branch node k:** If S^k_{LP} was pruned, then mark it as treated and continue to step 8. Otherwise, branch the node into new subproblems before marking S^k_{LP} as treated and then continue to step 8.

8. **Check for untreated nodes:** If there exist untreated nodes, return to step 4, else terminate as the best integer solution found so far is optimal for S.

4.2.2 Branch-and-cut

The branch-and-bound method is an exact solution method and will thus in theory always reach an optimal integer solution (given enough time, memory etc.). The branch-and-cut method is, as mentioned above, a generalization of the branch-and-bound, which in addition to the branching also can include extra inequalities (so-called cuts). These cuts may either be globally valid (such that they hold for all subproblems) or locally valid (subproblem specific cuts), such that the cuts do not change the optimal integer solution but hopefully speed up the convergence towards it. A simple cut that can be added is one based on the objective value, by adjusting the parameters in the problem S^k such that they are all integers, the optimal objective value for S^k must then also be integer[3]. If the optimal objective value to S^k_{LP} is

[3]Given that the problem is an ILP and not a MILP, in which case one should regard just the integer variable part of the objective function.

not an integer, i.e., $z^k_{\text{LP}} = \tilde{z} \notin \mathbb{Z}$, then the cut $z^k_{\text{LP}} = c^{\text{T}}y \geq \lceil \tilde{z} \rceil$ can be added to S^k to cut away this solution to S^k_{LP}, (while not changing the solution space of S^k.

4.3 Column generation

4.3.1 Non-integer linear optimization problems (LPs)

In this section we present the theory behind column generation in terms of Dantzig-Wolfe decomposition. Most of the theory here is based on [37, Ch. 10].

We will show how a general LP can be decomposed into master and subproblems, as a column generation problem. Consider the linear program (4.1) The constraints $Ax = b$ in (4.1) can be split into two sets of constraints, $A_1x = b_1$ and $A_2x = b_2$, with $A^{\text{T}} = [A_1^{\text{T}}, A_2^{\text{T}}]$, $A_1 \in \mathbb{R}^{m_1 \times n}$, $A_2 \in \mathbb{R}^{m_2 \times n}$ and $b^{\text{T}} = [b_1^{\text{T}}, b_2^{\text{T}}]$, $b \in \mathbb{R}^{m_1 \times 1}$, and $b \in \mathbb{R}^{m_2 \times 1}$. The problem (4.1) can then be rewritten as to

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & z = c^{\text{T}}x, \\ \text{subject to} \quad & A_1x = b_1, \\ & A_2x = b_2, \\ & x \geq 0, \end{aligned} \tag{4.3}$$

which can be expressed as the problem to

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & z = c^{\text{T}}x, \\ \text{subject to} \quad & A_1x = b_1, \end{aligned} \tag{4.4}$$

subject to the additional constraints

$$\begin{aligned} A_2x = b_2, \\ x \geq 0. \end{aligned} \tag{4.5}$$

The set of variables that fulfill (4.5) is a convex polyhedron, and any x feasible in (4.5) can be expressed as a convex combination of its extreme points and a positive linear combination of its extreme rays. This is mathematically expressed as for all $x \in \{x \in \mathbb{R}^n : A_2x = b_2, x \geq 0\}$ there exists λ, μ such that the following hold:

$$x = \sum_{i=1}^{L} \lambda_i u_i + \sum_{j=1}^{M} \mu_j v_j, \tag{4.6}$$

$$\sum_{i=1}^{L} \lambda_i = 1, \tag{4.7}$$

$$\lambda_i, \mu_j \geq 0, \qquad i \in \{1, \ldots, L\},\ j \in \{1, \ldots, M\}, \tag{4.8}$$

where u_i, $v_j \in \mathbb{R}^n$ are the extreme points and extreme rays, respectively, of the polyhedron defined by (4.5). The problem (4.1) can thus be written in terms of extreme points and extreme rays of (4.5), as the *master problem* to

$$\begin{aligned} \underset{\lambda,\mu}{\text{minimize}} \quad & z := \sum_{i:u_i\in\Omega} (c^{\mathrm{T}}u_i)\lambda_i + \sum_{j:v_j\in\Lambda} (c^{\mathrm{T}}v_j)\mu_j, \\ \text{subject to} \quad & \sum_{i:u_i\in\Omega} (A_1u_i)\lambda_i + \sum_{j:v_j\in\Lambda} (A_1v_j)\mu_j = b_1, \\ & \sum_{i:u_i\in\Omega} \lambda_i = 1, \\ & \lambda, \mu \geq 0, \end{aligned} \tag{MP}$$

where $\Omega = \{u_1, \ldots, u_L\}$ and $\Lambda = \{v_1, \ldots, v_M\}$.

For large LPs the sizes of Ω and Λ, i.e., the number of variables to consider (commonly referred to as columns[4]) can be so large that it is extremely hard to solve, and in some cases it can even be impossible to state all variables explicitly. However, while (MP) may have a very large number of columns, often only a relatively small number of them are needed to express an optimal solution. This property is what the column generation method exploits.

To handle the inconvenience with a very large number of variables in (MP) a smaller version of the problem with a subset of all variables is constructed. This problem is referred to as the *restricted master problem* (RMP) and is mathematically expressed as to

$$\begin{aligned} \underset{\lambda,\mu}{\text{minimize}} \quad & z := \sum_{i:u_i\in\tilde{\Omega}} (c^{\mathrm{T}}u_i)\lambda_i + \sum_{j:v_j\in\Lambda} (c^{\mathrm{T}}v_j)\mu_j, \\ \text{subject to} \quad & \sum_{i:u_i\in\tilde{\Omega}} (A_1u_i)\lambda_i + \sum_{j:v_j\in\tilde{\Lambda}} (A_1v_j)\mu_j = b_1, \\ & \sum_{i:u_i\in\tilde{\Omega}} \lambda_i = 1, \\ & \lambda, \mu \geq 0, \end{aligned} \tag{RMP}$$

where $\tilde{\Omega} \subseteq \Omega$ and $\tilde{\Lambda} \subseteq \Lambda$.

The idea of the column generation method is then to iteratively generate new columns to be added to (RMP) such that its optimal objective value is improved. When there are no more columns that can improve the objective value of RMP the optimal solution to RMP is also optimal in MP. This way one can find an optimal solution to the MP while only considering a hopefully relatively small number of columns in the corresponding RMP.

4.3.2 The subproblems

The efficiency of the column generation method is highly dependent on how fast new columns that improves the RMP-objective can be found and added to the problem. To be able to find these new columns for RMP we construct a so-called *subproblem*. The subproblem of RMP is to find an extreme point (u_i) or extreme ray (v_j) of (4.5) which is not yet in the set $\tilde{\Omega} \cup \tilde{\Lambda}$ and that would improve the optimal objective value of RMP the most.

[4]Variables in MP are often referred to as columns as each variable correspond to a separate column in the constraint matrix A_1.

To find the next variable to enter into RMP one can reason in terms of the simplex method. First let us rewrite MP in a more compact form. Let $\sum_{i:u_i\in\Omega}(c^{\mathrm{T}}u_i)\lambda_i + \sum_{j:v_j\in\Lambda}(c^{\mathrm{T}}v_j)\mu_j := [c^{\mathrm{T}}u \quad c^{\mathrm{T}}v]\begin{bmatrix}\lambda\\ \mu\end{bmatrix}$, and let $c_{\mathrm{MP}}^{\mathrm{T}} := [c^{\mathrm{T}}u \quad c^{\mathrm{T}}v]$ and $x_{\mathrm{MP}} := \begin{bmatrix}\lambda\\ \mu\end{bmatrix}$; the objective function of MP can then be rewritten as $z = c_{\mathrm{MP}}^{\mathrm{T}}x_{\mathrm{MP}}$. In a similar manner we rewrite the constraints of MP as $A_{\mathrm{MP}}^{\mathrm{T}}x_{\mathrm{MP}} = b_{\mathrm{MP}},\ x_{\mathrm{MP}} \geq 0$. Then MP can then be rewritten as to

$$\begin{aligned} \underset{x_{\mathrm{MP}}}{\text{minimize}} \quad & z = c_{\mathrm{MP}}^{\mathrm{T}}x_{\mathrm{MP}}, \\ \text{s.t.} \quad & A_{\mathrm{MP}}^{\mathrm{T}}x_{\mathrm{MP}} = b_{\mathrm{MP}}, \\ & x_{\mathrm{MP}} \geq 0. \end{aligned} \tag{MP compact}$$

With this compact form of the MP we can now directly apply the theory from section for the simplex method to get a criterion for what columns to generate for RMP. Consider $\bar{x}$ to be the optimal BFS for the current RMP, then $\bar{x}$ also corresponds to a BFS for MP (however not necessarily an optimal one). Imagine now that MP is solved using the simplex method, and that the method has reached the point where the best solution found is $x = \bar{x}$. Let x_B denote the basic variables for this solution, let the corresponding objective coefficients be c_B and let B be the corresponding part of $A_{\mathrm{MP}}^{\mathrm{T}}$. Similarly, let x_N denote its non-basic variables in x, and c_N and N the corresponding objective coefficients and part of $A_{\mathrm{MP}}^{\mathrm{T}}$. Then, given that $\bar{x}$ is not an optimal BFS for MP, the next step in the simplex solution scheme is to improve this solution by including a non-basic variable $(x_N)_j$, which has a negative reduced cost (i.e., such that $\tilde{c}_j := (c_{\mathrm{N}}^{\mathrm{T}} - c_{\mathrm{B}}^{\mathrm{T}}B^{-1}N)_j < 0$), to the set of basic variables. But to make this step in the simplex method for MP is equivalent to adding this non-basic variable $(x_N)_j$ to RMP and doing the same simplex step there. Thus, the criterion for a column to be added to RMP in order to improve the optimal solution to RMS, is that it has a negative reduced cost. So the subproblem of finding a column to add to RMP can be written as to

$$\underset{j\in\mathbb{N}:\ x_j\in\Omega\cup\Lambda}{\text{minimize}}\ \tilde{c}_j := (c_{\mathrm{N}}^{\mathrm{T}} - c_{\mathrm{B}}^{\mathrm{T}}B^{-1}N)_j.$$

The reduced cost is often written in terms of the dual variables of (RMP), which is also the form that it will be referred to in the remainder of this thesis. We therefore conclude this section by deriving this alternative expression for the reduced cost.

First let us have a quick reminder about duality theory for LPs: Any linear optimization problem (P) has a corresponding linear *dual problem* (D), such that, if (P) is written on the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & z_P := c^{\mathrm{T}}x, \\ \text{s.t.} \quad & A^{\mathrm{T}}x = b, \\ & x \geq 0, \end{aligned} \tag{P}$$

its linear dual problem (D) is to

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & z_D := b^{\mathrm{T}}\pi, \\ \text{s.t.} \quad & A\pi \leq c, \\ & \pi \text{ free}, \end{aligned} \tag{D}$$

.

and the optimal objective values of (P) and (D) are equal $z_P^* = c^{\mathrm{T}}x^* = b^{\mathrm{T}}\pi^* = z_D^*$, which also means that strong duality hold. Then, writing (RMP) and its dual in compact form we get,

$$\begin{aligned} \underset{x_{\mathrm{RMP}}}{\text{minimize}} \ & c_{\mathrm{RMP}}^{\mathrm{T}} x_{\mathrm{RMP}}, \\ \text{s.t.} \ & A_{\mathrm{RMP}}^{\mathrm{T}} x_{\mathrm{RMP}} = b_{\mathrm{RMP}}, \\ & x_{\mathrm{RMP}} \geq 0, \end{aligned}$$

(RMP compact)

$$\begin{aligned} \underset{\pi}{\text{maximize}} \ & b_{\mathrm{RMP}}^{\mathrm{T}} \pi, \\ \text{s.t.} \ & A_{\mathrm{RMP}} \pi \leq c_{\mathrm{RMP}}, \\ & \pi \text{ free}. \end{aligned}$$

(RMP dual)

We then have that $c_{\mathrm{RMP}}^{\mathrm{T}} x_{\mathrm{RMP}}^* = c_B^{\mathrm{T}} x_B = c_B^{\mathrm{T}} B^{-1} b = b^{\mathrm{T}} (B^{-1})^{\mathrm{T}} c_B = b^{\mathrm{T}} \pi^*$, where the third equality holds as $c_B^{\mathrm{T}} B^{-1} b$ is scalar and last equality holds due to strong duality (note that $(\pi^*)^{\mathrm{T}} = c_B^{\mathrm{T}} B^{-1}$. Then the reduced cost $\tilde{c}_j$ can be rewritten as

$$\tilde{c}_j := (c_N^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} N)_j = (c_{\mathrm{MP}}^{\mathrm{T}} - c_B^{\mathrm{T}} B^{-1} A_{\mathrm{MP}})_j = (c_{\mathrm{MP}}^{\mathrm{T}} - \pi^{*T} A_{\mathrm{MP}})_j, \tag{4.9}$$

So the subproblem of (RMP) is the problem of finding the column with lowest negative value of $\tilde{c}_j := (c_{\mathrm{MP}}^{\mathrm{T}} - \pi^{*T} A_{\mathrm{MP}})_j$, out of all columns in (MP).[5]

4.3.3 Integer linear optimization problems

In the previous section it was assumed that the problem was a continuous LP. In this case the column generation method will in theory eventually reach an optimal solution. However, if the problem would also have integer variables the column generation method can not be directly applied. This is because the column generation method must not have any integer variables in the master problem, or else the pricing of columns that builds upon LP duality and the simplex method will not work.

An ILP will still result in an integer master problem, even if the integer constraints on the variables is put into the subproblem. We will show this, without loss of generality, for a binary ILP. Let (S) be the integer problem we wish to solve using column generation. We then have that the problem to

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & z := c^{\mathrm{T}} x, \\ \text{subject to} \quad & A_1 x = b_1, \\ & A_2 x = b_2, \\ & x \in \{0, 1\}, \end{aligned} \tag{S}$$

[5]The optimal solution for the subproblem is thus the column that corresponds to the most violated constraint in the dual of the master problem (MP).

can be expressed as the problem to

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & z := c^{\mathrm{T}}x, \\ \text{subject to} \quad & A_1 x = b_1, \end{aligned} \tag{S_1}$$

subject to the additional constraints

$$\begin{aligned} A_2 x &= b_2, \\ x &\in \{0,1\}. \end{aligned} \tag{S_2}$$

Assume that the feasible set defined by (S_2) is bounded, and let $u_i \in \Omega$ be the extreme points of its convex hull (the convex hull being the smallest convex set containing the set at hand). Then it is true that any x that satisfy the constraints in (S_2) can be written as a convex combination of the points $u_i \in \Omega$. It is, however, not true that any convex combination of, $u_i \in \Omega$, will be a valid point for (S_2). To ensure that no convex combination of extreme points (i.e., columns in the master problem) is used such that the solution becomes infeasible in (S_2), a binary constraint on λ must be added, so that the master problem becomes to

$$\begin{aligned} \underset{\lambda,\mu}{\text{minimize}} \quad & z := \sum_{i:u_i \in \Omega} (c^{\mathrm{T}}u_i)\lambda_i, \\ \text{subject to} \quad & \sum_{i:u_i \in \Omega} (A_1 u_i)\lambda_i = b_1, \\ & \sum_{i:u_i \in \Omega} \lambda_i = 1, \\ & \lambda \in \{0,1\}^{|\Omega|}. \end{aligned} \tag{MP}$$

So we can see that placing all binary constraints of the original problem in the subproblem did not result in an LP master problem. However, even though the problem (S) can not be solved directly by using column generation, it is possible to make an LP relaxation of (S) (denoted S_{LP}) and generate columns for this problem instead. But as it is a relaxation of the original problem, its optimal value is only a lower bound on the optimal objective value of (S). While if the RMP is resolved with integer constraints (given all the generated columns from before) the optimal solution (if a solution exists) will just give an upper bound on the optimal value of (S). There is thus no guarantee that the method will reach an optimal solution to an ILP, but it will give an interval for its optimal objective value. However, it often been shown to generate good solutions, where the duality gap is never larger than the gap one gets from relaxing the integer constraints in the original problem. If there is no gap between the upper and lower bound of S, then the optimal solution found for S_{LP} is also optimal for S.

5

Column generation formulation

Problems A and B can be solved in the form that they are stated in Chapter 3 with some branch-and-cut method provided by optimization software, like CPLEX ([38, 39]). However, as the number of integer variables for these problems increase quickly for large numbers of vehicles and nodes, it soon becomes very time-consuming to solve the problems with a direct application of the branch-and-cut method. In this chapter we therefore present a column generation formulation of the two problems in the hope of improving the solution speed by decomposing the problems and solving their different parts separately.

The restricted master problem presented here is solved using the default solution methods of CPLEX, while the subproblems, so called elementary shortest path problems with resource constraints (ESPPRC) [40], are solved with a dynamic programming algorithm (Chapter 6.2) implemented in C++. Dynamic programming algorithms have been shown to solve many combinatorial problems really efficiently, among them shortest path problems with resource constraints [40, 41].

5.1 The master problem

In this section we present the master problem for problems A and B, with constraints regulating the number of vehicles used and the number of visits to customer nodes. The decomposition into master and subproblems is done in such way that all constraints being vehicle specific in the flow formulation enters the subproblems, while the remainder of the constraints forms the master problem. This results in a *set covering master problem* (which is easier to solve than if it was a set partitioning problem [42, pg. 108]), where each column represent a feasible route, and a shortest path subproblem with all constraints that defines a feasible route for the problem. The reason for this choice of decomposition is that it results in a simple master problem with few constraints, while, at the same time, yielding a type of subproblem that has been known to be solved rather efficiently using dynamic programming. By adding only vehicle specific constraints to the subproblem we can separate it into one problem for each vehicle. Further, as the vehicles are all identical, it is enough to solve only one of these subproblems to generate routes to the master problem.

Problem A and B only differs in how the recharging time is modelled, in terms of master problems this difference only show in the set of all feasible nodes (where the set of feasible routes for problem A is a subset of that set for problem B).

As problem A and B are MILPs we present the integer master problem (IMP), the non-integer master problem (MP), and the restricted master problem (RMP).

In Table 5.1 additional sets, parameters, and variables that are used to state the master problems are given.

Table 5.1: Additional sets, parameters, and variables introduced for the column generation formulation of Problem A and B

Notation	Description
Sets	
$\mathcal{R}$	The set of all feasible routes
Parameters	
c_r	Cost of route r (cost of propellant consumption) [€]
γ_{ir}	$= 1$ if route $r \in \mathcal{R}$ visits node $i \in \mathcal{V}$, $= 0$ otherwise
Variables	
λ_r	$= 1$ if route $r \in \mathcal{R}$ is used, $= 0$ otherwise
α	Dual variable corresponding to the constraint for the number of vehicles allowed in the master problem
β_i	Dual variable corresponding to the constraint for the number of visits to customer $i \in \mathcal{V}$

The routes in the master problem are defined by the sequence of nodes visited and the corresponding travelling cost.[1] As we regard a homogeneous fleet of vehicles the routes in $\mathcal{R}$ are not vehicle specific, instead they are all feasible for any vehicle.

In the IVP presented here each costumer must be visited at least once, and at most $F_{\max}$ vehicles can be used. Mathematically we present this problems as to

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \sum_{r\in\mathcal{R}} c_r\lambda_r \\ \text{s.t.} \quad & \sum_{r\in\mathcal{R}} \gamma_{ir}\lambda_r \geq 1, && i \in \mathcal{V}_{\text{cus}}, \\ & \sum_{r\in\mathcal{R}} \lambda_r \leq F_{\max}, \\ & \lambda_r \in \{0,1\}. && r \in \mathcal{R} \end{aligned} \tag{IMP}$$

To enable pricing of columns we present MP where the integrality of λ_r in IMP is relaxed. The subproblem solution then yields one or several new columns (i.e., vehicle routes) to add to the restricted master problem. The continuous master problem is thus defined as to

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \sum_{r\in\mathcal{R}} c_r\lambda_r, \\ \text{s.t.} \quad & \sum_{r\in\mathcal{R}} \gamma_{ir}\lambda_r \geq 1, && i \in \mathcal{V}_{\text{cus}} \\ & \sum_{r\in\mathcal{R}} \lambda_r \leq F_{\max}, \\ & \lambda_r \geq 0, && r \in \mathcal{R} \end{aligned} \tag{MP}$$

[1] The cost of the route depends on how much of the different propellants that was used when covering the distance.

and the corresponding LP dual problem is to

$$\begin{aligned} \underset{\alpha,\beta}{\text{maximize}} \quad & \sum_{i\in\mathcal{V}_{\text{cus}}} \beta_i + \alpha F_{\max}, \\ \text{s.t.} \quad & \sum_{i\in V_c} \beta_i\gamma_{ir} + \alpha \le c_r, && r \in \mathcal{R} \\ & \alpha \le 0, \beta_i \ge 0. && i \in \mathcal{V}_{\text{cus}} \end{aligned} \tag{MP-dual}$$

Finally, the restricted master problem (RMP) is then to

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \sum_{r\in\widetilde{\mathcal{R}}} c_r\lambda_r, \\ \text{s.t.} \quad & \sum_{r\in\widetilde{\mathcal{R}}} \gamma_{ir}\lambda_r \ge 1, && i \in \mathcal{V}_{\text{cus}} \\ & \sum_{r\in\widetilde{\mathcal{R}}} \lambda_r \le F_{\max}, \\ & \lambda_r \ge 0, && r \in \widetilde{\mathcal{R}} \end{aligned} \tag{RMP}$$

where $\widetilde{\mathcal{R}} \subseteq \mathcal{R}$. The RMP is solved in terms of its LP dual, the optimal variable values of which are used in the subproblem.

5.2 Subproblems

As stated before, in the subproblems we put all constraints from the flow formulations that are vehicle specific. Each subproblem then becomes a kind of shortest path problem with capacity constraints, which can be solved efficiently using dynamic programming algorithms like in [40]. The subproblems then generates feasible routes that are sent, in terms of new columns, to the restricted master problem. In this section we present (I) subproblem A, where the recharging time is constant and thus independent of the battery level, and (II) subproblem B where the recharging time depends linearly on the battery level.

Note that all constraints that we choose to include in the subproblems are vehicle specific. Then, as the objective function of the subproblems, which in accordance to Section 4.3.2 becomes (sA1), is a linear function of costs for each vehicle, the subproblem can be split into $F_{\max}$ separate subproblems (one for each vehicle). As problems A and B both consider a homogeneous vehicle fleet, all these vehicle specific problems are in fact identical. Thus it is enough to solve only one of these problems as they all will yield the same solutions.

Next we give the expressions for subproblems A and B.

Subproblem A: Find the cheapest vehicle route when the recharging time is constant

The objective function (sA1) in this subproblem corresponds to the reduced cost for a route r in the master problem (i.e., $\tilde{c}_r := (c_{\text{MP}}^{\text{T}} - \pi^{*T} A_{\text{MP}})_r$ as defined in (4.9)). The reduced cost for this problem consists of three parts: (1) the cost of propellant (diesel and electricity) needed to travel the route; (2) a negative contribution (a "bonus")

of $-\beta_i$ for visiting customer node i (β_i, $i \in \mathcal{V}_{\text{cus}}$, are variables from (MP-dual), while we let $\beta_i = 0$ for $i \in \mathcal{V} \setminus \mathcal{V}_{\text{cus}}$); (3) a constant term $-\alpha$, which is also a variable from (MP-dual). The constraints (sA2) says that each node may be visited at most once, while (sA3) assures that if the vehicle arrives at a node it must also leave the node. The time constraints (sA4)–(sA6) assures that no sub-tours are made, that the vehicle arrives in time to nodes and that all routes starts and ends at the depot. The constraints (sA7) and (sA8) treat the cargo balance (the cargo decreases when serving a customer node) and the propellant constraints (sA9)–(sA11) assure that enough propellant is used to cover the route, and that the amount of battery used does not exceed the amount the vehicle has. Finally (sA12)–(sA14) keeps all variable values positive and adds binary constraints on the x-variables.

$$\underset{x,y,z^{\text{Fu}},z^{\text{El}},\tau,u}{\text{minimize}} \quad \widetilde{c} := \sum_{(i,j)\in\mathcal{A}} (c^{\text{Fu}} z_{ij}^{\text{Fu}} + c^{\text{El}} z_{ij}^{\text{El}} - \beta_i x_{ij}) - \alpha, \tag{sA1}$$

subject to

$$\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij} \leq 1, \qquad i \in \mathcal{V}, \tag{sA2}$$

$$\sum_{j\in\mathcal{V}:(j,i)\in\mathcal{A}} x_{ji} = \sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij}, \qquad i \in \mathcal{V}, \tag{sA3}$$

$$e_i \leq \tau_i \leq l_i, \qquad i \in \mathcal{V}, \tag{sA4}$$

$$\tau_i \leq T_{\max} - (t_{i0} + p_i), \qquad i \in \mathcal{V} \setminus \{0\}, \tag{sA5}$$

$$\tau_j \geq \tau_i + (t_{ij} + p_i) - M(1 - x_{ij}), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}, \tag{sA6}$$

$$u_0 \leq U_{\max}, \tag{sA7}$$

$$u_j \leq u_i - q_i x_{ij} + U_{\max}(1 - x_{ij}), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}, \tag{sA8}$$

$$x_{ij} d_{ij} = r^{\text{Fu}} z_{ij}^{\text{Fu}} + r^{\text{El}} z_{ij}^{\text{El}}, \qquad (i,j) \in \mathcal{A}, \tag{sA9}$$

$$y_j \leq y_i - z_{ij}^{\text{El}} + Q(1 - x_{ij}), \qquad i \in \mathcal{V}_{\text{cus}}, j \in \mathcal{V}, \tag{sA10}$$

$$y_j \leq Q - z_{ij}^{\text{El}} + Q(1 - x_{ij}), \qquad i \in \mathcal{V} \setminus \mathcal{V}_{\text{cus}}, j \in \mathcal{V}, \tag{sA11}$$

$$x_{ij} \in \{0, 1\}, \qquad (i,j) \in \mathcal{A}, \tag{sA12}$$

$$z_{ij}^{\text{El}}, z_{ij}^{\text{Fu}}, \geq 0, \qquad (i,j) \in \mathcal{A}, \tag{sA13}$$

$$\tau_i, y_i, u_i \geq 0, \qquad i \in \mathcal{V}. \tag{sA14}$$

Subproblem B: Find the cheapest vehicle route when the recharging time depends on the battery level

The objective function (sB1) in subproblem B is again the reduced cost for the column r. The constraints (sB2) allow the vehicle to arrive at each node at most once, (sB3) says that if a vehicle visits a node it must also leave it. The time constraints (sB4)–(sB6) are the same as for subproblem A for non-recharging nodes, while (sB7)–(sB9) are similar time constraints regulating the time spent at each recharging node. The constraints (sB10)–(sB11) treat the cargo balance and (sB12)–(sB14) treat the propellant balance. Lastly (sB15)–(sB17) makes all variables positive and

adds binary constraints to the x-variables.

$$\underset{x,y,z^{\mathrm{Fu}},z^{\mathrm{El}},s,u,\tau}{\text{minimize}} \quad \widetilde{c} := \sum_{(i,j)\in\mathcal{A}} (c^{\mathrm{F}} z_{ij}^{\mathrm{Fu}} + c^{\mathrm{El}} z_{ij}^{\mathrm{El}} - \beta_i x_{ij}) - \alpha, \tag{sB1}$$

subject to

$$\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij} \leq 1, \qquad i \in \mathcal{V}, \tag{sB2}$$

$$\sum_{j\in\mathcal{V}:(j,i)\in\mathcal{A}} x_{ji} = \sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij}, \qquad i \in \mathcal{V}, \tag{sB3}$$

$$e_i \leq \tau_i \leq l_i, \qquad i \in \mathcal{V}, \tag{sB4}$$

$$\tau_i \leq T_{\max} - (t_{i0} + p_i), \qquad i \in \mathcal{V}_{\mathrm{cus}}, \tag{sB5}$$

$$\tau_j \geq \tau_i + (t_{ij} + p_i) - M(1 - x_{ij}), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V} \setminus \mathcal{V}_{\mathrm{rec}}, \tag{sB6}$$

$$\tau_i \leq T_{\max} - (t_{i0} + s_i), \qquad i \in \mathcal{V} \setminus \{0\}, \tag{sB7}$$

$$\tau_j \geq \tau_i + s_i + t_{ij} - M(1 - x_{ij}), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}_{\mathrm{rec}}, \tag{sB8}$$

$$s_i \geq g(Q - y_i), \qquad i \in V_r, \tag{sB9}$$

$$u_0 \leq U_{\max}, \tag{sB10}$$

$$u_j \leq u_i - q_i x_{ij} + U_{\max}(1 - x_{ij}), \qquad j \in \mathcal{V} \setminus \{0, i\}, i \in \mathcal{V}, \tag{sB11}$$

$$x_{ij} d_{ij} = r^{\mathrm{Fu}} z_{ij}^{\mathrm{Fu}} + r^{\mathrm{El}} z_{ij}^{\mathrm{El}}, \qquad (i,j) \in \mathcal{A}, \tag{sB12}$$

$$y_j \leq y_i - z_{ij}^{\mathrm{El}} + Q(1 - x_{ij}), \qquad i \in \mathcal{V}_{\mathrm{cus}}, j \in \mathcal{V}, \tag{sB13}$$

$$y_j \leq Q - z_{ij}^{\mathrm{El}} + Q(1 - x_{ij}), \qquad i \in \mathcal{V} \setminus \mathcal{V}_{\mathrm{cus}}, j \in \mathcal{V}, \tag{sB14}$$

$$x_{ij} \in \{0, 1\}, \qquad (i,j) \in \mathcal{A}, \tag{sB15}$$

$$z_{ij}^{\mathrm{El}}, z_{ij}^{\mathrm{Fu}} \geq 0, \qquad (i,j) \in \mathcal{A}, \tag{sB16}$$

$$\tau_i, s_j, y_i, u_i \geq 0, \qquad i \in \mathcal{V}, j \in \mathcal{V}_{\mathrm{rec}}. \tag{sB17}$$

The objective in each of the two subproblems is to find the route that correspond to the most violated constraint in the problem (MP-dual), given the optimal dual variable values of (RMP). The rest of the constraints are similar to the ones presented in the VRP flow formulation of problem a presented earlier, with the restriction that only one vehicle may be used and not all costumers have to be visited.

Also, observe that even though the objective of the subproblems is to minimize the reduced cost function (i.e., $\tilde{c} := (c_{\mathrm{MP}}^{\mathrm{T}} - \pi^{*T} A_{\mathrm{MP}})_r$), this is not the cost of the route that is sent back to the RMP. The cost corresponding to a route in RMP is the actual cost of travelling the route ($c_r := \sum_{(i,j)\in\mathcal{A}: x_{ij}^r = 1} (c^{\mathrm{Fu}} z_{ij}^{\mathrm{Fu}} + c^{\mathrm{El}} z_{ij}^{\mathrm{El}})$), i.e., the cost of the propellant used for the route.

We have now presented the column generation formulation for both Problem A and B. In the next chapter we present a way to solve our hybrid VRPs when they are given in this form. Our main focus lays in solving the shortest path problems using a dynamic programming algorithm.

6
Solving the hybrid-VRP

In this section we present some details on how to solve the hybrid VRPs presented by means of column generation. The solution scheme applied for the hybrid-VRPs using column generation is illustrated in Figure 6.1.

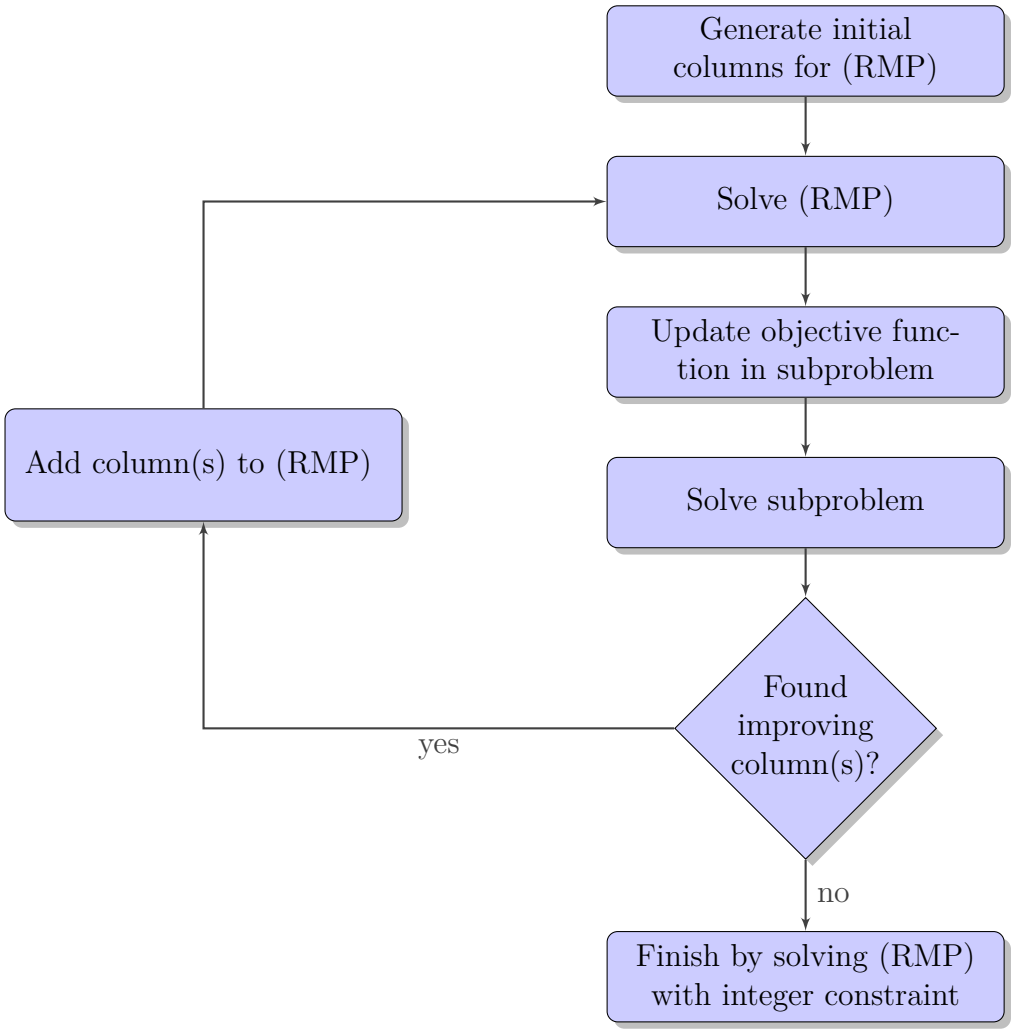

Figure 6.1: The solution scheme used to solve the hybrid VRP with column generation

The solution scheme in Figure 6.1 starts by using a simple greedy heuristic to

attain a feasible starting solution for the RMP. Then RMP is solved and the subproblem's objective function is updated with the new optimal dual variable values of RMP. After that the subproblem is solved in order to find columns that can improve the optimal solution of the RMP; if such columns are found, they are sent to RMP. Once no more columns can be found for the RMP, the algorithm is terminated after solving RMP with integer requirements on the variables (as it is for the IMP). The solving of RMP without integer requirements gives a lower bound on the optimal value of IMP, and RMP with integer requirements gives an upper bound of it.

Solving the subproblem is often the most time consuming step in the column generation procedure. However, as it is not necessary to return the optimal route in each iteration it is possible to reduce the time spent solving subproblems by terminating these early. Another common way to speed up the solution process is to send more than one column (if available) to the RMP each time the subproblem is solved [43]. This way the number of columns generated increases faster, which could result in a faster convergence to an optimal solution to the MP.

For our problems we apply these ideas with the following rules. A column is sent to the restricted master problem while solving a subproblem if

- its reduced cost of a found column is negative, and
- it has the best (lowest) reduced cost of all columns checked for this iteration of the column generation procedure.

Further, the solution of the subproblem is terminated if either of the following two criteria hold

1. Ten new columns have been sent to the RMP in the present column generation iteration
2. At least one new column has been sent to the RMP and the time it has taken to solve the subproblem has past a given upper time limit.

6.1 Solving the subproblems

Both subproblem A and B are elementary shortest path problems with resource constraints (ESPPRC). Resources in these problems being time, cargo, and battery, all being limited by constraints in the subproblem. The solution method that will be used to solve these problems is a so called *labeling algorithm*, similar to the one presented in [40]. This method and its application to subproblems A and B are presented in this section. However, first we introduce some new notations and formally define our use of the terms *path* and *route*.

Definition 1. A path is an ordered set of nodes.

Definition 2. An elementary path is a path without subtours, i.e., such that no node (except for the depot) is visited more than once.

Let $p_{il} := \{0, l_1, l_2, \ldots, i\}$ denote a path that starts in the depot (i.e., node 0) and

ends in node i, and where the notation l_n denotes the nth node in path l. For convenience we let a path that starts and ends in the depot (denoted p_{0l}) be referred to as a route.

Definition 3. A node j is unreachable from path p_{il} if p_{il} is not allowed to visit this node next. There are four different reasons that would make a node j unreachable for p_{il}:

1. Node j is in the path p_{il} ($j \in p_{il}$)
2. The vehicle would arrive to the node after its time window defined in problem A and B.
3. There is not enough cargo left to meet the demand of the node that is specified in the problem.
4. A visit to the node would result in an arrival to the depot which is later than the latest return time specified by the problem.

Definition 4. Let $\mathbb{T}_{il} := \{T_{il}^1, \ldots, T_{il}^w\}$ denote the set of resources, $\mathbb{V}_{il} := \{V_{il}^1, \ldots, V_{il}^n\}$ be the set of variables that shows the reachability of nodes (where $V_{il}^j = 1$ if node $j \in \mathcal{V}$ is unreachable, and $V_{il}^j = 0$ otherwise), and let c_{il} denote the reduced cost of the path. We then define the *label* of a path p_{il} as $\mathcal{L}_{il} := (\mathbb{T}_{il}, \mathbb{V}_{il}, c_{il}) = (T_{il}^1, \ldots, T_{il}^w, V_{il}^1, \ldots, V_{il}^n, c_{il})$.

With these definitions we can describe a *labeling algorithm* as associating labels $\mathcal{L}_{il}$ to paths p_{il} and then extending these paths. When no more labels can be created the algorithm terminates, and the optimal path is the one with the lowest reduced cost in the label at the end node.

The labeling algorithm used in this report make use of *dominance criteria*, which decrease the number of paths to be extended. Dominance of a path occures if the dominance criterion holds true. When a path p_{i1} dominates p_{i2}, the path p_{i2} will no longer be considered in the solution procedure.

We here give an example of dominance criterion that can be used when solving a SPPRC, when T_{il}^k is the amount of resource k that has been used for path l until the visit of node i. The dominance criterion says that path p_{i1} dominates p_{i2} if the following dominance inequalities hold:

$$T_{i1}^k \leq T_{i2}^k, \qquad k \in \{1, \ldots, w\}, \tag{D.1}$$

$$V_{i1}^j \leq V_{i2}^j, \qquad j \in \mathcal{V}, \tag{D.2}$$

$$c_{i1} \leq c_{i2}, \tag{D.3}$$

For this dominance criterion it has been proven ([40]) that Claim 1 holds true.

Claim 1. It is enough to consider only the non-dominated paths in the label setting algorithm in [40] to be able to guarantee optimality for the solution of a SPPRC where the T_{il}^k denotes the use of resources. ■

With the terms given in this section we will in the next section present the dynamic programming algorithm used to solve our subproblems.

6.1.1 The dynamic programming algorithm

Now some last expressions that are needed before presenting the dynamic programming algorithm. After which the labeling algorithm is presented in pseudo code.

- **Successor node:** A successor node to $i \in \mathcal{V}$ is a node $j \in \mathcal{V}$ such that $(i, j) \in \mathcal{A}$, i.e., it is possible to travel directly from i to j.
- **Extension of labels:** A label $\mathcal{L}_{il}$ corresponding to the path $p_{il} = \{0, l_1, l_2, \ldots, i\}$ is said to be extended to node j if a new label is created at node j corresponding to the path $p_{jl} = \{0, l_1, l_2, \ldots, i, j\}$.
- **Treated labels:** A label $\mathcal{L}_{il}$ is said to be treated if it is extended to all possible successors of node i.
- **Efficient labels:** A label $\mathcal{L}_{il}$ is said to be efficient with respect to a node i if it is not dominated by any label in the node.

Setup for the algorithm:

- Λ_i: Set of non-treated efficient labels at node i
- $\hat{\Lambda}_i$: Set of treated efficient labels in node i
- $\Gamma(i)$: Set of successors of node i
- $\mathcal{L}_0$: Starting label for the depot node
- ***Extend***$(\mathcal{L}_{il}, j)$: The label resulting from an extension of $\mathcal{L}_{il}$ to node j, $(\mathcal{L}_{il}, j) = \varnothing$ if the extension is infeasible (which cannot be an efficient label).
- ***Dominated***(Λ_j)/***Dominated***$(\hat{\Lambda}_j)$: The set of non-treated/treated labels of node j that are no longer efficient.

Algorithm 2 Solving Subproblem

```
1: /* Initialization */
2: Λ_0 ⟵ {L_0};
3: for i ∈ V \ {0} do
4:     Λ_i ⟵ ∅;
5: end for
6:
7: /* Labeling algorithm */
```

8: **while** $\bigcup_{i=0}^{n} \Lambda_i \neq \emptyset$ **do**
9: **for** $i \in V$ **do**
10: **if** $\Lambda_i \neq \emptyset$ **then**
11: **Choose** $\mathcal{L} \in \Lambda_i$
12: **for** $j \in \Gamma(i)$ **do**
13: **if** (**Extend** $(\mathcal{L}, j)$ efficient label of node j) **then**
14: $\Lambda_j \longleftarrow \Lambda_j \cup$ **Extend**$(\mathcal{L}, j)$;
15: $\Lambda_j \longleftarrow \Lambda_j \backslash$**Dominated**$(\Lambda_j)$;
16: $\hat{\Lambda}_j \longleftarrow \hat{\Lambda}_j \backslash$**Dominated**$(\hat{\Lambda}_j)$;
17: **end if**
18: **end for**
19: $\Lambda_i \longleftarrow \Lambda_i \setminus \{\mathcal{L}\}$;
20: $\hat{\Lambda}_i \longleftarrow \hat{\Lambda}_i \cup \{\mathcal{L}\}$;
21: **end if**
22: **end for**
23: **end while**

Algorithm 2 in words:

- Initialize by adding a "0-path label"[1] in the depot as an untreated label, and define all other nodes to be empty (containing no labels).
- As long as untreated paths exist, repeat the following:
  - Chose a node $i \in \mathcal{V}$ and pick an untreated label from it (if no such labels exist, chose another node and repeat).
  - Loop through all successors of node i and check if the chosen label can be extended to these nodes (if the extended label is efficient in the new node). If the extended label is dominated by an existing label it is not added to the node. Else, add the extended label to the set of untreated labels and remove any labels dominated by the new label.
  - Move the treated label from the set of untreated labels in node i to the set of treated labels in node i.

This algorithm can be used in simplicity without any dominance criteria, which would result in a brute force solution method (enumerating all possible solutions). This approach is, however, of course a very inefficient way of solving problems, being impractical to use even for very small problem instances. But when some dominance criteria, like (D.1)–(D.3), is used it results in far fewer paths to consider and the solution speed of Algorithm 2 would thus greatly increase. For subproblem A the criteria (D.1)–(D.3) translate into

[1]With 0-path label means a label representing a vehicle before the start of a route, with all of its resources at their respective optimal levels; in our case time = 0, the battery is fully charged, and the cargo is filled to its capacity.

$$time(\mathcal{L}_{i1}) \leq time(\mathcal{L}_{i2}) \tag{6.1}$$
$$cargo(\mathcal{L}_{i1}) \geq cargo(\mathcal{L}_{i2}) \tag{6.2}$$
$$battery(\mathcal{L}_{i1}) \geq battery(\mathcal{L}_{i2}) \tag{6.3}$$
$$V_{i1}^{j} \leq V_{i2}^{j}, \qquad j \in \mathcal{V} \tag{6.4}$$
$$c_{i1} \leq c_{i2} \tag{6.5}$$

which would be enough to guarantee an optimal solution from Algorithm 2. It is however possible to strengthen the dominance criteria further for our hybrid problems, which is highly desirable, since it may speed up the solution process of the subproblem even further. Therefore, we next present our stronger dominance criteria for subproblems A and B and prove that Claim 1 holds even for these criteria.

6.2 Dominance criteria for subproblems A and B

It is necessary to have a consistent rule for creating labels in Algorithm 2. The labels must also hold enough information about the path, such that valid dominance criteria can be applied and the optimal path is not lost. For subproblems A and B it is not obvious how to construct such labels. We next present the idea for creating these labels for a path p_{il}; more details are given in Sections 6.2.1 and 6.2.3.

For problem A it is always preferable to leave each node as early as possible; hence the departure time of node i should be the earliest possible. The use of battery does not affect the travelling time but it allow the vehicle to cover distances cheaply; it is therefore always preferable to use as much battery as possible when traveling. Therefore, the cost of the path should be as low as possible the lowest cost possible —the battery level is a direct result of this cost. The amount of cargo is a direct result of the path, and the unreachable nodes result from the path taken and the departure time at node i.

In subproblem B, however, there is a trade-off between a low cost and an early departure time from node i; this is a result of the recharging times being dependent on the battery charge level on arrival at a recharging node. So using more battery will keep the cost relatively low, but to the expense of more time spent at the recharging nodes, and vice versa. To solve this dilemma we store the following information in the labels: Save the lowest possible cost for the path, which defines the battery level as well as the departure time from node i. In order to keep the information about the earliest possible departure time for path p_{il}, save also the information of how much the departure time could be improved. This is done by defining a resource *flex*, which is how much the departure time can be improved by spending less time recharging the battery. The cargo is a direct result of the path taken, and the set of unreachable nodes results from the path taken and the earliest possible departure time from node i.

These rules for creating labels provide the information needed to define our dominance criteria for subproblems A and B.[2]

Before we present the dominance criteria for subproblem A, recall the notations defined in Tables 3.1 and 5.1.

6.2.1 Labels in subproblem A

In accordance to the description of our labels in the previous section we let the labels for subproblem A contain the following information:

Table 6.1: Description of the labels for subproblem A

Description	Descriptive label	Notation
Lowest possible cost for path l at node i [€]	$cost(\mathcal{L}_{il})$	c_l^i
Earliest possible departure time from node i [h]	$time(\mathcal{L}_{il})$	t_l^i
Battery level at departure from node i [kWh] corresponding to the lowest possible cost	$battery(\mathcal{L}_{il})$	b_l^i
Amount of cargo at departure from node i [kg]	$cargo(\mathcal{L}_{il})$	q_l^i
Set of unreachable nodes from node i when path l has been used	$unreachable(\mathcal{L}_{il})$	Ω_{il}

The battery level in the label $\mathcal{L}_{il}$ is a result of the driving pattern for path l that results in the cost c_l^i, and the amount of cargo depends on what customer nodes were visited.

6.2.2 Dominance criteria for subproblem A

Definition 5. The following inequalities must hold if label $\mathcal{L}_{i1}$ dominates $\mathcal{L}_{i2}$:

$$q_1^i \geq q_2^i, \tag{C.1}$$
$$c_1^i \leq c_2^i - \max\{0, H(b_2^i - b_1^i)\}, \tag{C.2}$$
$$t_1^i \leq t_2^i, \tag{C.3}$$
$$\Omega_{i1} \subseteq \Omega_{i2}, \tag{C.4}$$

where $H := (c^{\mathrm{Fu}} r^{\mathrm{Fu}} / r^{\mathrm{El}} - c^{\mathrm{El}}) > 0$. The inequality (C.1) says that there is more cargo left for path 1 compared to path 2. The inequality (C.2) guarantees lower cost for all extensions of path 1 compared to those of path 2. Inequality (C.3) assures that the departure time from node i is earlier for path 1 than for path 2. Finally, inequality (C.4) says that if a node is unreachable from node i, if attained via path 1, it is unreachable also if node i is attained via path 2.

[2] Subproblems A and B can be solved using using the dominance criterion (D.1)–(D.3) combined with the labels described in Section 6.2. The dominance criteria in Section 6.2 are stronger than (D.1)-(D.3), and are therefore likely to speed up Algorithm 2 (see the comparison in Chapter 7).

The parameter H denotes the extra cost associated with covering a distance using diesel instead of using electricity. For example, if a distance could be fully covered using electricity with an electricity consumption of Δb kWh, then covering the same distance using diesel would cost €$H\Delta b$ more. Dominance inequality (C.2) therefore says that the cost of path 1 is so low that even if path 2 results in a higher battery level, it is not possible get a better cost from extending path 2 than for the same extension of path 1. Example 6.2.1 illustrates how this dominance criterion is stronger than the criterion (D.1)–(D.3).

Example 6.2.1. Consider the scenario in Figure 6.2 where two different paths are compared, both starting in the depot d and visiting customers c_1 and c_2, but path 2 also visiting a recharging station r. The different lines mark what kind of propellant that was used to cover that distance; a dotted/solid line illustrates the distance

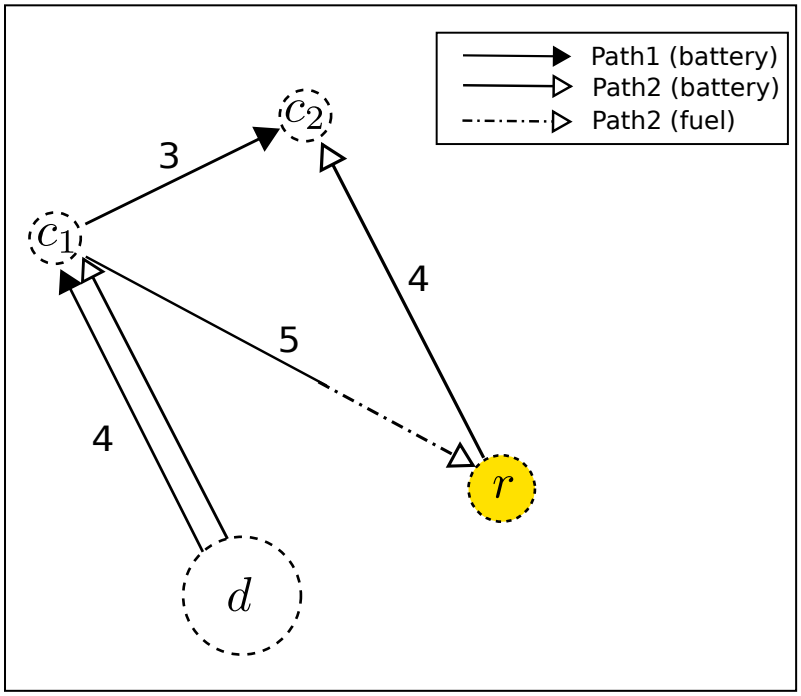

Figure 6.2: A scenario of two different paths starting in the depot node d and ending in the costumer node c_2. The number written next to each arc denotes its length (in km)

traveled by conventional fuel/electricity. We now wish to investigate whether path 1 can be a better start than is path 2 for any route passing through customer nodes c_1 and c_2. To be able to compare path 1 and 2, let t be the time it takes to travel one unit of distance and t_r be the time for recharging. Further, let b be the capacity of the battery and assume that the battery is empty for the vehicle taking path 1 when it reaches node c_2 (which means that a vehicle on path 2 travels on fuel for two units of distance between the nodes c_1 and r). Moreover, let c_E (c_F) be the cost for traveling one unit of distance using electricity (by fuel). We compare the two routes by using the information given to construct–for each of the two paths–the labels in node c_2.

$\mathcal{L}_{c_2 1}$:

$$
\begin{aligned}
unreachable(p_1) &= \{c_1, c_2\} \\
cost(p_1) &= (4+3)c_E = 7c_E \\
time(p_1) &= (4+3)t = 7t \\
battery(p_1) &= 0
\end{aligned}
$$

$\mathcal{L}_{c_2 2}$:

$$
\begin{aligned}
unreachable(p_2) &= \{c_1, c_2, r\} \\
cost(p_2) &= 11c_E + 2c_F \\
time(p_2) &= 13t + t_r \\
battery(p_2) &= \frac{3}{7}b
\end{aligned}
$$

From the computed labels it follows that $unreachable(p_1) \subseteq unreachable(p_2)$, $cost(p_1) \leq cost(p_2)$, and $time(p_1) \leq time(p_2)$, which indicate that p_1 dominates p_2, but in terms of battery level we have that $battery(p_1) \leq battery(p_2)$.

The question is now whether a path extended from path 2 can have a lower cost than the same extension of path 1 (as a result of the higher battery level resulting from path 2)? The maximal cost improvement that path p_2 can have relative to path p_1 is $3(c_F - c_E)$ which is sufficient for resulting in a better overall cost if and only if $3(c_F - c_E) > 4c_E + 2c_F$, which in turn holds if and only if it holds that $c_F > 7c_E$. Hence, provided that $c_F \leq 7c_E$ any path that extends p_1 will always be better than the same path extension of p_2, this means that p_1 dominates p_2. Which shows that dominance criterion (C.1)–(C.4) gives domination of paths when the criterion (D.1)–(D.3) does not (as the later requires $battery(p_1) \geq battery(p_2)$ for domination).

Claim 2. For subproblem A it is enough to regard only non-dominated labels, given dominance criteria (C.1)–(C.4), to find the optimal route using Algorithm 2.

Proof. The claim will be proved by showing that any extension of a dominated label to a feasible node will always be dominated by a label of some other path. Specifically, if $\mathcal{L}_{i1}$ dominates $\mathcal{L}_{i2}$, then the extension of p_{i2} to node $j \notin \Omega_{i2}$ will be dominated by the extension of p_{i1} to this node. The proof treats the different criteria one at the time.

Criterion (C.1) Prove that the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies q_1^j \geq q_2^j \tag{6.6}$$

holds true.

Subproof 1. According to the definitions in Section 6.2.1, the cargo at departure from node i is q_1^i for path 1 and q_2^i for path 2. Let d_j be the demand of node j. The cargo left after serving node j is then $q_1^j := q_1^i - d_j$ and $q_2^j := q_2^i - d_j$, respectively. It follows that

$$q_1^j = q_1^i - d_j \geq q_2^i - d_j = q_2^j,$$

which proves implication (6.6). □

Criterion (C.2) Prove the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies c_1^j \leq c_2^j - \max\{0, H(b_2^j - b_1^j)\} \tag{6.7}$$

where $H = c^{\text{Fu}} r^{\text{Fu}} / r^{\text{El}} - c^{\text{El}} > 0$.

Subproof 2. Let c_{ij} denote the cheapest cost of travelling from node i to node j for path 1 (given its battery level in node i). The lowest cost for path 2 to travel the same distance may be lower than c_{ij} if a larger part of the distance can be covered using electricity compared to path 1. Let $\varepsilon \geq 0$ denote how much can be gained (in cost) by travelling the arc (i, j) in the way done for path 2 compared to path 1. If it is not cheaper to travel arc (i, j) for path 2 compared to path 1 then $\varepsilon = 0$, else ε denotes that cost difference. With these notations the cost for paths 1 and 2, respectively, extended by node j, fulfill $c_1^j = c_1^i + c_{ij}$ and $c_2^j \geq c_2^i + c_{ij} - \varepsilon$, where $\max\{0, H(b_2^i - b_1^i)\} \geq \varepsilon \geq 0$. The new difference in battery level in node j equals in the case of a non-recharging node $\max\{0, b_2^j - b_1^j\} = \max\{0, b_2^i - b_1^i\} - \dfrac{\varepsilon}{H} \geq 0$, or in the case of a recharging nodes, $\max\{0, b_2^j - b_1^j\} = 0$. It follows that

$$\begin{aligned} c_1^j &= c_1^i + c_{ij} \\ &\leq c_2^i + c_{ij} - H \max\{0, (b_2^i - b_1^i)\} \\ &= c_2^i + c_{ij} - \varepsilon - H \left(\max\{0, (b_2^i - b_1^i)\} - \frac{\varepsilon}{H} \right) \\ &\leq c_2^j - \max\{0, H(b_2^j - b_1^j)\}. \end{aligned}$$

The first equality holds per definition, while the second line is true as $H \geq 0$ (it is cheaper to drive on electricity than diesel). The third row is a simple rewriting of the second row. Finally, the last inequality holds as $\max\left\{0, b_2^j - b_1^j\right\} = \max\left\{0, b_2^i - b_1^i\right\} - \dfrac{\varepsilon}{H}$ when j is not a recharge node and $\max\left\{0, b_2^i - b_1^i\right\} - \dfrac{\varepsilon}{H} \geq 0$ for the case when j is a recharging node. □

Criterion (C.3) Prove that the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies t_1^j \leq t_2^j \tag{6.9}$$

holds true.

Subproof 3. Let t_{ij} denote the time it takes to travel from node i to node j, d_j the service time in node j and e_j let the earliest visit time for node j. The departure times for the two paths at node j can then be written as $t_1^j = \max\{e_j, t_1^i + t_{ij}\} + d_j$ and $t_2^j = \max\{e_j, t_2^j + t_{ij}\} + d_j$. It follows that

$$t_1^j = \max\{e_j, t_1^i + t_{ij}\} + d_j \leq \max\{e_j, t_2^i + t_{ij}\} + d_j = t_2^j$$

where the inequality holds as $t_1^i \leq t_2^i$ due to dominance in node i. □

Criterion (C.4) Prove that the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies \Omega_{j1} \subseteq \Omega_{j2} \tag{6.10}$$

holds true.

Subproof 4. The set Ω_{il} of unreachable nodes in label $\mathcal{L}_{il}$ can be decomposed into three different sets of nodes, as $\Omega_{il} = \Omega_{il}^{v} \cup \Omega_{il}^{tw} \cup \Omega_{il}^{d}$, where Ω_{il}^{v} denotes the set of nodes which has already been visited by the path, Ω_{il}^{tw} denotes the set of nodes that can not be reached within the time windows for the respective nodes, and Ω_{il}^{d} denotes the set of nodes for which a visit would result in a too late return to the depot. Comparing these subsets of unreachable nodes yields:

$$\begin{aligned}
&\Omega_{j1}^{v} = \Omega_{i1}^{v} \cup \{j\} \subseteq \Omega_{i1} \cup \{j\} \subseteq \Omega_{i2} \cup \{j\} \subseteq \Omega_{j2} \\
&\Omega_{j1}^{tw} \subseteq \Omega_{j2}^{tw} \\
&\Omega_{j1}^{d} \subseteq \Omega_{j2}^{d}
\end{aligned}$$

where the second inclusion follows since it holds that $t_1^i + t_{ij} \leq t_2^i + t_{ij}$ and the third since $t_1^j \leq t_2^j$ holds. It then follows that

$$\Omega_{j1} = \Omega_{j1}^{v} \cup \Omega_{j1}^{tw} \cup \Omega_{j1}^{d} \subseteq \Omega_{j2} \cup \Omega_{j2}^{tw} \cup \Omega_{j2}^{d} = \Omega_{j2},$$

which proves implication (6.10). □

Thus it has been shown that "$\mathcal{L}_{i1}$ dominates $\mathcal{L}_{i2}$" $\implies$ "$\mathcal{L}_{j1}$ dominates $\mathcal{L}_{j2}$" and hence we have proven Claim 2. ■

6.2.3 Labels in subproblem B

In Problem B the recharging time is dependent on the battery level at arrival in the recharging node. Therefore, using less electricity before the recharging decreases the time spent at the recharging node. This trade-off between time and cost should be taken into account when creating the dominance criteria (in problem A there is no trade-off between time and cost). The labels for problem B must thus also contain information on the time that can be gained at a higher cost. Each label $\mathcal{L}_{il}^{\mathrm{B}}$ in subproblem B contains the same cost, cargo and set of unreachable node information as the labels for problem A has. In addition the labels contain:

Table 6.2: Description of the additions to the labels for subproblem B

Description	Descriptive label	Notation
Departure time corresponding to the "best cost alternative" [h]	$time(\mathcal{L}_{il})$	t_l^i
How much departure time can be improved at the price of a higher cost [h]	$flex(\mathcal{L}_{il})$	f_l^i
Battery at departure for the best cost alternative [kWh]	$battery(\mathcal{L}_{il})$	b_l^i

In the case without time windows *flex* represents the accumulated time spent recharging, which one can avoid by driving more on diesel.[3] The trade–off between time and cost is affine and we can thus express the departure time as a linear function of cost $t(c) = kc + t_0$, for some constants $k < 0$ and $t_0 \geq 0$. This trade-off is illustrated in Example 6.2.2.

The coefficient k can be expressed in terms of the recharging rate g, the propellant costs c^{El} and c^{Fu}, and the propellant consumption rates r^{El} and r^{Fu}, according to the following. Covering some distance using Δb kWh of battery corresponds to $\Delta t = g\Delta b$ more time spent at the next recharging node, and $\Delta c = -H\Delta b$ saved compared to the same distance covered using diesel (where $H = (c^F r^F / r^E - c^E) > 0$ represents the cost of replacing one unit of battery usage with diesel). The coefficient of the time-cost trade-off lines can thus be written as $k = \Delta t / \Delta c = -g/H$.

Example 6.2.2. The trade-off between cost and time Consider a label $\mathcal{L}^{\mathrm{B}}_{i1}$ (corresponding to a path p_{i1}) with $cost(\mathcal{L}^{\mathrm{B}}_{i1}) = c^i_1$, $time(\mathcal{L}^{\mathrm{B}}_{i1}) = t^i_1$ and $flex(\mathcal{L}^{\mathrm{B}}_{i1}) = f^i_1$. This means that the lowest cost possible for a vehicle to travel the path p_{il} is c^i_1, and given this cost the vehicle can depart from node i at the earliest at the time t^i_1. However, the departure time can be earlier if less time is spent recharging the battery. As f^i_1 denotes the total time spent recharging the battery, the earliest possible departure from node i (when traveling the path p_{i1}) is $t^i_1 - f^i_1$, corresponding to the case when p_{i1} is travelled only on diesel. But to reduce the travel time in this way the vehicle must cover larger distances than necessary using diesel, which will increase the cost of the route. This time-cost trade-off is illustrated in Figure 6.3.

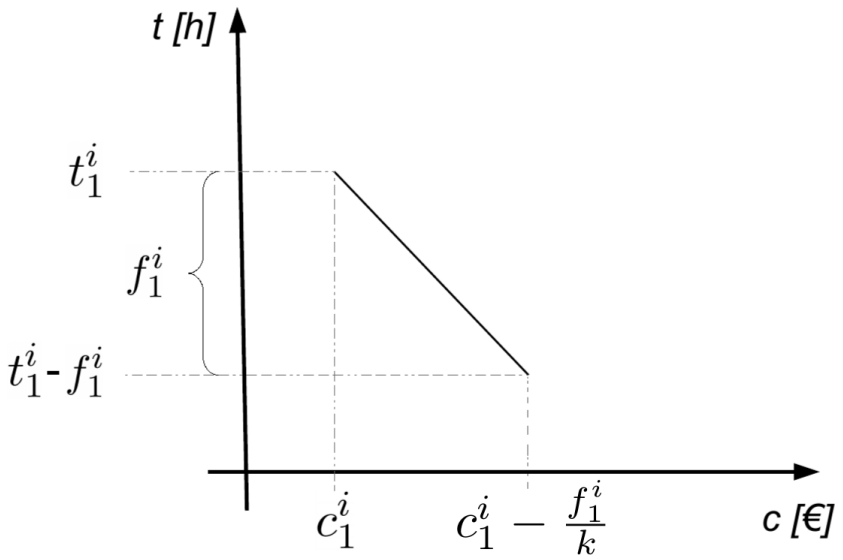

Figure 6.3: A visualisation of the trade-off between cost and time in the label $\mathcal{L}_{i1}$.

The solid line in Figure 6.3 is the part of the line $t(c) = kc + t_0$ that represents the possible time-cost combinations for path p_{i1}. The cost that corresponds to the earliest possible departure time, $t^i_1 - f^i_1$, becomes $c^i_1 - f^i_1/k$ where $k = -g/H$.

[3] When time windows are considered, otherwise feasible departure times can be made infeasible, which may decrease the amount of time flex.

6.2.4 Dominance criteria for Problem B without time windows

The tighter the time (i.e., resource) constraints in the ESPPRC are, the more efficient can the algorithm in [40] be expected to solve it. The reason for heavily constrained problems being easier to solve is that tighter constraints typically reduce the number of feasible solutions. This algorithm is therefore not likely to perform well as applied to Problem B without time windows; hence we will not report solutions to any of these instances. Nevertheless, we present dominance criteria for solving Problem B without time windows but with a maximum duration limit of each route. The reason for this is twofold: The dominance criteria clarify the concept of the parameter *flex* in the label. It also clarifies why dominance criteria for Problem B (with time windows) must compare the battery levels in the labels directly, and not indirectly as for Problem A.

For problem B without time windows the following dominance criteria must hold if label $\mathcal{L}^{\mathrm{B}}_{i1}$ dominates label $\mathcal{L}^{\mathrm{B}}_{i2}$:

$$q^i_1 \geq q^i_2, \tag{C.5}$$
$$c^i_1 \leq c^i_2 - \max\{0, H(b^i_2 - b^i_1)\}, \tag{C.6}$$
$$t^i_1 - f^i_1 \leq t^i_2 - f^i_2 - \max\{0, g(b^i_2 - b^i_1)\}, \tag{C.7}$$
$$t^i_1 - kc^i_1 \leq t^i_2 - kc^i_2 - \max\{0, g(b^i_2 - b^i_1)\}, \tag{C.8}$$
$$\Omega_{i1} \subseteq \Omega_{i2}. \tag{C.9}$$

with $\quad H = c^F r^F / r^E - c^E > 0 \quad$ and $\quad k = -g/H < 0.$

Dominance criteria (C.5) and (C.9) are equivalent to the corresponding criteria for Problem A (there denoted (C.1) and (C.4)). Criteria (C.6)–(C.8) assure that for any possible "cost-time combination" of $\mathcal{L}_{i2}$ there exists a "cost-time combination" for $\mathcal{L}_{i1}$ in which both cost and time are better (lower). The "max" operators further assures that this would still be the case if both paths (p_{i1} and p_{i2}) where extended in the same way to other nodes.

In Figure 6.4, criteria (C.6)–(C.8) are illustrated in a time-cost trade-off plot; in the area AC criteria (C.6) and (C.8) hold, in BC criteria (C.7) and (C.8) hold and in area ABC all three criteria hold. For path 1 (p_{i1}) to dominate path 2 it is then enough that dominance criteria (C.5) and (C.9) hold and that path 1's trade-off line lays either (at least partly) in the area ABC or both partly in AC and partly in BC (see Example 6.2.3).

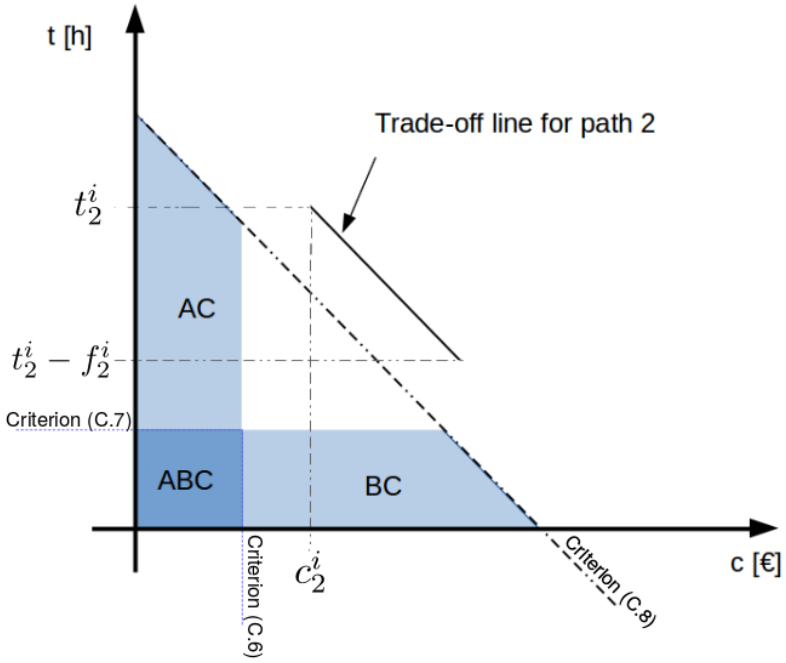

Figure 6.4: Illustration of the regions where dominance criteria (C.6)–(C.8) hold for domination of L_{i2}, and the time-cost trade-off line for label $\mathcal{L}_{i2}$

Example 6.2.3. Assume that the relations $q_1^i \geq q_2^i$ and $\Omega_{i1} \subseteq \Omega_{i2}$ hold and that the battery level of $\mathcal{L}_{i1}$ is greater than or equal to that of $\mathcal{L}_{i2}$, i.e., that $\max\{0, b_2^i - b_1^i\} = 0$. Figure 6.5 illustrates criteria (C.6)–(C.8) for some values $t_1^i, c_1^i, \ldots$, in such a case, in which $\mathcal{L}_{i1}$ dominates $\mathcal{L}_{i2}$ as the line for p_{i1} lies in both regions AC and BC, which means that criteria (C.6)–(C.8) are fulfilled.

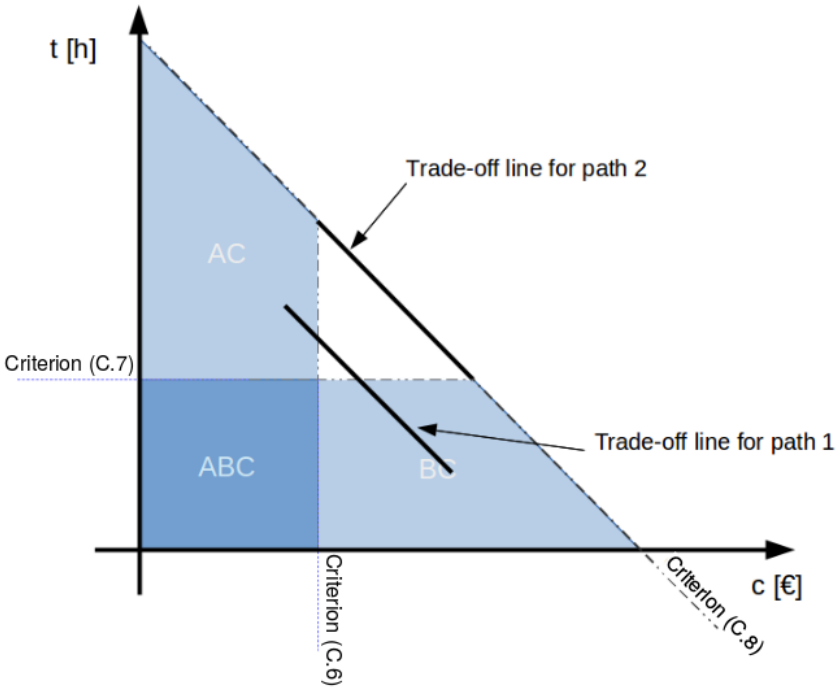

Figure 6.5: Illustration of the criteria (C.6)–(C.8) when $\max\{0, b_2^i - b_1^i\} = 0$.

Assume instead that $\max\{0, b_2^i - b_1^i\} = \Delta b > 0$; this scenario is illustrated in Figure 6.6. Now, criterion (C.6) is no longer satisfied, i.e., $c_1^i > c_2^i - H(b_2^i - b_1^i)$. This means that the potential cost improvement of p_{i2} relative to p_{i1}, is so large that it can not be guaranteed that all extensions of p_{i1} will have lower costs than all corresponding extensions of p_{i2}. Hence, $\mathcal{L}_{i1}$ will not dominate $\mathcal{L}_{i2}$.

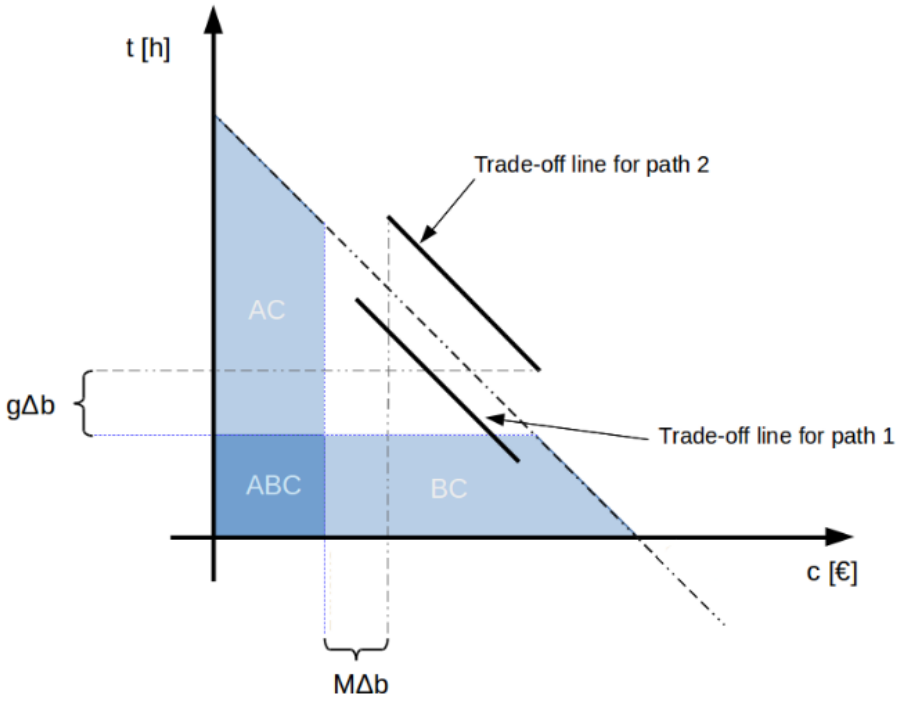

Figure 6.6: Illustration of the criteria (C.6)–(C.8) when $\max\{0, b_2^i - b_1^i\} = \Delta b > 0$.

Given these new dominance criteria Claim 3 holds.

Claim 3. When solving subproblem B without time windows it is enough to consider only non-dominated paths as defined by the dominance criteria (C.5)–(C.9).

Proof. We will show—by extending the dominating path analogously to the dominated path that a dominated path—can only possess dominated child paths. Let p_{i1} denote the dominating path in node i and p_{j1} the extension of the path p_{i1} to node j, and analogously p_{i2} the dominated path and p_{j2} its extension.

For proofs of Criteria (C.5) and (C.6) see the proof for the dominance criteria of subproblem A.

Criterion (C.7)
Prove that the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies t_1^j - f_1^j \leq t_2^j - f_2^j - \max\{0, g(b_2^j - b_1^j)\} \tag{6.9}$$

holds.

Subproof 1. The best possible departure time from node i for p_{il} is $t_l^i - f_l^i$, as t_l^i is the departure time corresponding to the lowest cost and f_l^i denotes the largest possible decrease (at additional cost) of this departure time. The earliest possible departure time from node j for the two paths 1 and 2 can then be expressed as

$$t_1^j - f_1^j = t_1^i - f_1^i + t_{ij} + d_j$$

and

$$t_2^j - f_2^j \geq t_2^i - f_2^i + t_{ij} + d_j - \varepsilon,$$

respectively, where d_j denotes the service time for path 1 at node j (observe that d_j is battery dependent if j is a recharging node), t_{ij} is the traveling time of arc (i, j), and ε denotes the difference—between paths 1 and 2—of the time spent in node j ($\varepsilon = 0$ whenever j is not a recharging node). Note that the inequality

$$\max\{0, g(b_2^j - b_1^j)\} + \varepsilon = \max\{\varepsilon, g(b_2^j - b_1^j)\} \leq \max\{0, g(b_2^i - b_1^i)\}$$

hold. This inequality comes from the fact that the battery difference between two paths can not increase when traveling the same arc, which in turn gives us the relation $\varepsilon \leq \max\{0, g(b_2^i - b_1^i)\}$. Hence, with this information, the inequalities

$$\begin{aligned} t_1^j - f_1^j &= t_1^i - f_1^i + t_{ij} + d_j \\ &\leq t_2^i - f_2^i + t_{ij} + d_j - \max\{0, g(b_2^i - b_1^i)\} \\ &\leq t_2^i - f_2^i + t_{ij} + d_j - \varepsilon - \max\{0, g(b_2^j - b_1^j)\} \\ &\leq t_2^j - f_2^j - \max\{0, g(b_2^j - b_1^j)\} \end{aligned}$$

hold, and implication (6.9) is proven. □

Criterion (C.8)
Prove that the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies t_1^j - kc_1^j \leq t_2^j - kc_2^j - \max\{0, g(b_2^j - b_1^j)\} \tag{6.10}$$

holds.

Subproof 2. Define Δb_{travel} to be how much more battery path 2 uses on arc (i, j) compared to path 1. Also define $\Delta b_{\text{recharge}}$ to be how much more kWh that path 1 charges in node j compared to path 2. Note that $\Delta b_{\text{travel}} = 0$ if path 2 does not use more battery than path 1 over arc (i, j), and $\Delta b_{\text{recharge}} = 0$ if path 1 does not spend more time recharging—compared to path 1—at node j or if node j is not a recharging node. Given these definitions it holds that

$$\max\{0, b_2^i - b_1^i\} = \max\{0, b_2^j - b_1^j\} + \Delta b_{\text{travel}} + \Delta b_{\text{recharge}}. \tag{6.13}$$

By denoting the cost of traveling arc (i, j) for path 1 as c_{ij}, the traveling time as t_{ij} and its service time in node j as d_j, the lowest cost for path 1 and path 2 to node j can then be written as

$$c_1^j = c_1^i + c_{ij} \tag{6.14a}$$

$$c_2^j \geq c_2^i + c_{ij} - H\Delta b_{\text{travel}} \tag{6.14b}$$

respectively. The corresponding departure times can be written as

$$t_1^j = t_1^i + t_{ij} + d_j \tag{6.15a}$$
$$t_2^j = t_2^i + t_{ij} + d_j - g\Delta b_{\text{recharge}}. \tag{6.15b}$$

Hence, the relation

$$\begin{aligned}
t_2^j - t_1^j - k(c_2^j - c_1^j) &\geq t_2^i + t_{ij} + d - g\Delta b_{\text{recharge}} - (t_1^i + t_{ij} + d) - k(c_2^j - c_1^j) \\
&= t_2^i - t_1^i - g\Delta b_{\text{recharge}} - k(c_2^j - c_1^j) \\
&\geq t_2^i - t_1^i - g\Delta b_{\text{recharge}} - k(c_2^i + c_{ij} - H\Delta b_{\text{travel}} - (c_1^i + c_{ij})) \\
&= t_2^i - t_1^i - g\left(\Delta b_{\text{recharge}} + \Delta b_{\text{travel}}\right) - k(c_2^i - c_1^i) \\
&= t_2^i - t_1^i - g\left(\max\{0, b_2^i - b_1^i\} - \max\{0, b_2^j - b_1^j\}\right) - k(c_2^i - c_1^i) \\
&\geq g\max\{0, b_2^i - b_1^i\} - g\left(\max\{0, b_2^i - b_1^i\} - \max\{0, b_2^j - b_1^j\}\right) \\
&= g\max\{0, b_2^j - b_1^j\}
\end{aligned}$$

hold, which concludes the proof of (6.10). The two first inequality comes from insertion of the expressions (6.14a),(6.14b),(6.15a), and (6.15b). The third equality results from (6.13), and the last inequality is true due to the dominance criterion (C.8) in node i. □

Criterion (C.9)
Prove the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies \Omega_{j1} \subseteq \Omega_{j2}. \tag{6.14}$$

Subproof 3. The set Ω_{il} of unreachable nodes in label $\mathcal{L}_{il}$, for Problem B without time windows, can be decomposed into two different sets, as $\Omega_{il} := \Omega_{il}^v \cup \Omega_{il}^d$; where Ω_{il}^v denotes the set of nodes already visited in the path, and Ω_{il}^d denotes the set of nodes for which a visit would result in a too late return to the depot. Then we have

$$\begin{aligned}
&\Omega_{j1}^d \subseteq \Omega_{j2}^d \\
&\Omega_{j1}^v = \Omega_{i1}^v \cup \{j\} \subseteq \Omega_{i1} \cup \{j\} \subseteq \Omega_{i2} \cup \{j\} \subseteq \Omega_{j2}
\end{aligned}$$

where the first inclusion is a result of the fact that $t_1^j - f_1^j \leq t_2^j - f_2^j$. It follows that

$$\Omega_{j1} = \Omega_{j1}^v \cup \Omega_{j1}^d \subseteq \Omega_{j2} \cup \Omega_{j2}^d = \Omega_{j2},$$

which proves the implication (6.17). □

This concludes the proof of Claim 3. ■

6.2.5 Dominance criteria for Problem B with time windows

For subproblem B with time windows we must in addition to the criteria for the case without time windows also include a criterion concerning the battery levels. We then have the following criteria that must hold if label $\mathcal{L}_{i1}$ shall dominate $\mathcal{L}_{i2}$:

$$b_1^i \geq b_2^i, \tag{C.10}$$
$$q_1^i \geq q_2^i, \tag{C.11}$$
$$c_1^i \leq c_2^i, \tag{C.12}$$
$$t_1^i - f_1^i \leq t_2^i - f_2^i, \tag{C.13}$$
$$t_1^i - kc_1^i \leq t_2^i - kc_2^i, \tag{C.14}$$
$$\Omega_{i1} \subseteq \Omega_{i2}, \tag{C.15}$$

with $k = -g/H < 0$ and $H = c^F r^F / r^E - c^E > 0$.

Criterion (C.10) requires a dominating path to have more battery than the path it is dominating. Criteria (C.11)–(C.15) are the same criteria as (C.5)–(C.9) under the assumption of a larger battery level for the dominating path. The reason why the battery criterion must be added when considering time windows (in addition to the depot's time window) is illustrated in Example 6.2.4.

Example 6.2.4.
Consider the scenario in Figure 6.7, where Criteria (C.6)–(C.8) hold, and assume that even criteria (C.11) and (C.15) hold. This is enough for label 1 ($\mathcal{L}_{i1}$) to dominate label 2 ($\mathcal{L}_{i2}$) when there are no time windows. Assume now that node i has an associated time window such that serving of the node is not possible before the earliest possible arrival time of path p_{i2}.

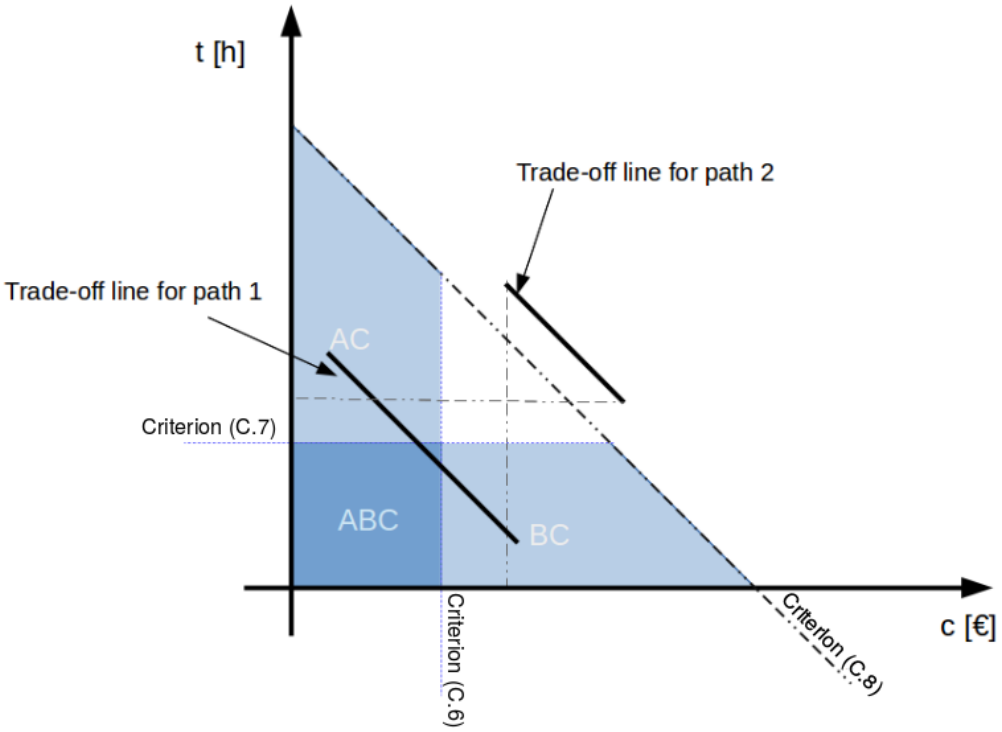

Figure 6.7: Dominance criteria (C.6)–(C.8) without time windows.

Figure 6.8 shows how this time window has shortened the trade-off line for path p_{i1}, so that the best departure times for the paths p_{i1} and p_{i2} are the same. If label $\mathcal{L}_{i2}$ has a higher battery level than $\mathcal{L}_{i1}$ (i.e. $b_1^i < b_2^i$), the result could be an extensions of $\mathcal{L}_{i2}$ having better (earlier) departure times than extensions of $\mathcal{L}_{i2}$ (in the case a recharging node is visited after node i). However, if we add the criterion $b_1^i \geq b_2^i$, then extensions of $\mathcal{L}_{i1}$ are still guaranteed to have departure times at least as early as for the corresponding extensions of $\mathcal{L}_{i2}$.

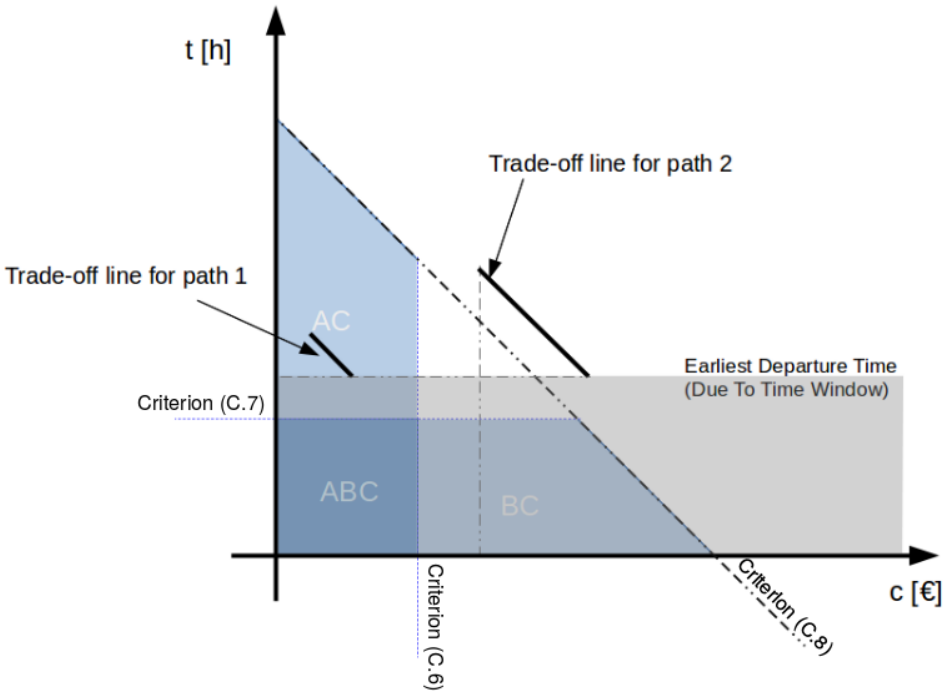

Figure 6.8: Dominance criteria (C.6)–(C.8) with time windows.

Example 6.2.4 shows the need of Criteria (C.10) for Problem B. Next we will show that these dominance criteria are enough for Algorithm 2 to return an optimal solution.

Claim 4. When solving subproblem B with time windows it is enough to consider only non-dominated paths as defined by the dominance criteria (C.10)–(C.15).

Proof. Let p_{i1} denote the dominating path in i and p_{j1} the extension of this path p_{i1} to node j, and analogously p_{i2} the dominated path and p_{j2} its extension. The proof for Criterion (C.11) is identical to that of the corresponding criterion in Problem A, and is therefore omitted here. The proof of Criterion (C.15) differs from that of Problem A only in the expression $t_1^j - f_1^j$ (rather than t_1^j) of the earliest departure time from node j; hence, it is also left out here.

Criterion (C.10)
Prove the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies b_1^j \geq b_2^j. \tag{6.17}$$

Subproof 1. The proof of (6.17) is trivial when node j is a recharging node, we therefore assume this is not the case. Define Δb to be the amount of battery needed to cover the entire arc (i,j) fully on battery. The battery levels for path 1 and path 2 in node j are then $b_1^j = b_1^i - \min\{\Delta b, b_1^i\}$, and $b_2^j = b_2^i - \min\{\Delta b, b_2^i\}$, as the battery level in the label correspond to the cheapest cost of the paths.

In the case $\min\{\Delta b, b_1^i\} = b_1^i$ it implies that $\min\{\Delta b, b_2^i\} = b_2^i$ as $\Delta b \geq b_1^i \geq b_2^i$, and hence $b_1^j = b_2^j = 0$. If, instead, $\min\{\Delta b, b_1^i\} = \Delta b$ it implies that $b_1^j = b_1^i - \Delta b \geq \min\{0, b_2^i - \Delta b\} = b_2^i - \min\{b_2^i, \Delta b\} = b_2^j$, which concludes the proof of (6.17). □

Criterion (C.12)
Prove the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies c_1^j \leq c_2^j. \tag{6.18}$$

Subproof 2. Define c_{ij} to be the cost for path 1 to travel arc (i,j), and $c_{ij} - \varepsilon$ to be the cost of path 2 to travel arc (i,j), where $\varepsilon \geq 0$ as $battery(\mathcal{L}_{i1}) \geq battery(\mathcal{L}_{i2})$. The cost of the paths p_{j1} and p_{j2} can then be written as

$$c_1^j = c_1^i + c_{ij} - \frac{1}{k}\max\{0, t_1^i + t_{ij} - l_j\}$$

and

$$c_2^j = c_2^i + c_{ij} + \varepsilon - \frac{1}{k}\max\{0, t_2^i + t_{ij} - l_j\},$$

where the "$-1/k \max\{\dots\}$" terms is the cost that must be sacrificed to reach node j within its time window.

Extending $c_1^j - c_2^j$ now gives us

$$\begin{aligned} c_1^j - c_2^j &= c_1^i - (c_2^i + \varepsilon) - \frac{1}{k}\Big(\max\{0, t_1^i + t_{ij} - l_j\} - \max\{0, t_2^i + t_{ij} - l_j\}\Big) \\ &= \begin{cases} c_1^i - c_2^i - \varepsilon - \frac{1}{k}\cdot 0, & t_1^i, t_2^i \leq l_j - t_{ij}, \quad (6.19a) \\ c_1^i - c_2^i - \varepsilon - \frac{1}{k}\cdot(-(t_2^i + t_{ij} - l_j)), & t_1^i \leq l_j - t_{ij} \leq t_2^i, \quad (6.19b) \\ c_1^i - c_2^i - \varepsilon - \frac{1}{k}\cdot(t_1^i + t_{ij} - l_j), & t_1^i \geq l_j - t_{ij} \geq t_2^i, \quad (6.19c) \\ c_1^i - c_2^i - \varepsilon - \frac{1}{k}\cdot(t_1^i - t_2^i), & t_1^i, t_2^i \geq l_j - t_{ij}. \quad (6.19d) \end{cases} \end{aligned}$$

The expression in (6.19a) is less than zero since $c_1^i \leq c_2^i$ and $\varepsilon \geq 0$. For (6.19b) it is true that $t_2^i + t_{ij} - l_j \geq 0$ and thus $-\frac{1}{k}\cdot(-(t_2^i + t_{ij} - l_j)) \leq 0$, so

$$c_1^i - c_2^i - \varepsilon - \frac{1}{k}\cdot(-(t_2^i + t_{ij} - l_j)) \leq c_1^i - c_2^i - \varepsilon \leq 0.$$

In (6.19c) we want to show that $c_1^i - c_2^i - \varepsilon - \frac{1}{k}\cdot(t_2^i + t_{ij} - l_j) \leq 0$, which is the same thing as proving $k(-c_1^i + c_2^i + \varepsilon) + t_1^i + t_{ij} - l_j \leq 0$. We then get

$$\begin{aligned} k(-c_1^i + c_2^i + \varepsilon) + t_1^i + t_{ij} - l_j &\leq k(-c_1^i + c_2^i + \varepsilon) + t_1^i + t_{ij} - l_j - (t_2^i + t_{ij} - l_j) \\ &= k(c_2^i - c_1^i) - (t_2^i - t_1^i) + k\varepsilon \\ &\leq k\varepsilon \leq 0, \end{aligned}$$

where the first inequality holds because $\max\{0, t_2^i + t_{ij} - l_j\} = 0$, the second due to dominance in node i giving $t_1^i - kc_1^i \leq t_2^i - kc_2^i$, and the final inequality is valid since $k \leq 0$. Finally, for (6.19d) the dominance criteria $t_1^i - kc_1^i \leq t_2^i - kc_2^i$ we get

$$c_1^i - c_2^i - \varepsilon - \frac{1}{k} \cdot (t_1^i - t_2^i) \leq c_1^i - c_2^i - \varepsilon - \frac{1}{k} \cdot k(c_1^i - c_2^i) = -\varepsilon \leq 0.$$

Hence, we have proven the implication (6.18). □

Criterion (C.13)
Prove the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies t_1^j - f_1^j \leq t_2^j - f_2^j. \tag{6.20}$$

Subproof 3. It has been shown for Criterion (C.8) that the implication (6.20) holds for Problem B without time window in node j. Let $\hat{f}_1^j$ and $\hat{f}_2^j$ denote the flex of the paths p_{j1} and p_{j2}, respectively, in the case of no time window in node j.If the earliest visit time of node j (e_j) is later than earliest arrival time of path 1 to node j, then the earliest departure time possible for path 1 from node j would be $e_j + d_j$, where d_j is the service time at j for the path. The earliest departure times for path 1 and 2 from node j in the presents of time windows then becomes

$$t_1^j - f_1^j = \max\{e_j + d_j, t_1^j - \hat{f}_1^j\}$$

and

$$t_1^2 - f_2^j = \max\{e_j + d_j + \varepsilon, t_2^j - \hat{f}_2^j\}$$

respectively, where $d_j + \varepsilon$ with $\varepsilon \geq 0$ is the service time for path 2 (possibly longer than the service time for path 1 due to battery difference). As it holds that $t_1^j - \hat{f}_1^f \leq t_2^j - \hat{f}_2^j$ it follows that

$$t_1^j - f_1^j = \max\{e_j + d_j, \hat{t}_1^j - \hat{f}_1^j\} \leq \max\{e_j + d_j + \varepsilon, \hat{t}_2^j - \hat{f}_2^j\} = t_2^j - f_2^j,$$

which proves implication (6.20). □

Criterion (C.14)
Prove the implication

$$\mathcal{L}_{i1} \text{ dominates } \mathcal{L}_{i2} \implies t_1^j - kc_1^j \leq t_2^j - kc_2^j. \tag{6.21}$$

Subproof 4. Let t_{ij} denote the time it takes to travel the arc (i, j), and recall that the time window for node j is $[e_j, l_j]$. Further, let c_{ij} denote the cheapest cost—given the battery level left in node i—for path 1 to cover the arc (i, j), the lowest cost for path p_{j1} then becomes

$$c_1^j = c_1^i + c_{ij} - \frac{1}{k} \max\{0, t_1^i + t_{ij} - l_j\}, \tag{6.22}$$

where the term $\frac{1}{k} \max\{0, t_1^i + t_{ij} - l_j\}$ is the cost of the improvement in arrival time that must be made to reach node j in time (i.e. the cost of used flex). The, for this cost, corresponding departure time of path p_{1j} becomes

$$t_1^j = t_1^i + t_{ij} + d_j + \max\{0, e_j - (t_1^i + t_{ij})\} - \max\{0, t_1^i + t_{ij} - l_j\}, \tag{6.23}$$

where $\max\{0, e_j-(t_1^i+t_{ij})\}$ is how much too early node j is reached, and $\max\{0, t_1^i + t_{ij} - l_j\}$ is how much the arrival time must be decreased for the vehicle to arrive in time to node j. The expression $t_1^j - kc_1^j$ can then be rewritten as

$$\begin{aligned} t_1^j - kc_1^j &= t_1^i + t_{ij} + d_j + \max\{0, e_j - (t_1^i + t_{ij})\} - \max\{0, t_1^i + t_{ij} - l_j\} \\ &\quad - k(c_1^i + c_{ij} - \frac{1}{k}\max\{0, t_1^i + t_{ij} - l_j\}) \\ &= t_1^i + t_{ij} + d_j + \max\{0, e_j - (t_1^i + t_{ij})\} - k(c_1^i + c_{ij}) \\ &= \begin{cases} t_1^i + t_{ij} + d_j + 0 - k(c_1^i + c_{ij}), & t_1^i \geq e_j - t_{ij}, \\ e_j + d_j - k(c_1^i + c_{ij}), & t_1^i \leq e_j - t_{ij}, \end{cases} \end{aligned} \tag{6.24}$$

where the first equality comes from the insertion of (6.22) and (6.23), and where the last equality gives the two possible cases of the max term. Similarly, with $d_j + \varepsilon_1$ as the service time for path 2 in node j and $c_{ij} + \varepsilon_2$ as the cost for path 2 to travel arc (i, j) (with[4] $\varepsilon_1 \geq 0, \varepsilon_2 \geq 0$), $t_2^j - kc_2^j$ can be rewritten as

$$\begin{aligned} t_2^j - kc_2^j &= t_2^i + t_{ij} + (d_j + \varepsilon_1) + \max\{0, e_j - (t_2^i + t_{ij})\} - k(c_2^i + (c_{ij} + \varepsilon_2)) \\ &= \begin{cases} t_2^i + t_{ij} + (d_j + \varepsilon_1) - k(c_2^i + (c_{ij} + \varepsilon_2)), & t_2^i \geq e_j - t_{ij}, \\ e_j + d_j - k(c_2^i + (c_{ij} + \varepsilon_2)), & t_2^i \leq e_j - t_{ij}. \end{cases} \end{aligned} \tag{6.25}$$

Finally we can prove (6.21) by rewriting the expression $(t_1^j - kc_1^j) - (t_2^j - kc_2^j)$ by the use of (6.24) and (6.25), and show that it is less than zero. This giving the expression

$$\begin{aligned} &(t_1^j - kc_1^j) - (t_2^j - kc_2^j) = \\ &= \begin{cases} t_1^i - t_2^i - k(c_1^i - c_2^i - \varepsilon_2) - \varepsilon_1 & \leq k\varepsilon_2 - \varepsilon_1 \leq 0 \\ t_1^i - t_2^i - k(c_1^i - c_2^i) + t_2^i + t_{ij} - e_j + k\varepsilon_2 & \leq t_2^i + t_{ij} - e_j + k\varepsilon_2 \leq k\varepsilon_2 \leq 0 \\ e_j - t_2^i - t_{ij} - \varepsilon_1 - k(c_1^i - c_2^i - \varepsilon_2) & \leq -k(c_1^i - c_2^i) + k\varepsilon_2 - \varepsilon_1 \leq 0 \\ k(c_2^i - c_1^i + \varepsilon_2) & \leq 0 \end{cases} \end{aligned}$$

The first row corresponds to $\max\{0, e_j - (t_1^i + t_{ij})\} = \max\{0, e_j - (t_2^i + t_{ij})\} = 0$. The first inequality holds as $t_1^i - kc_1^i \leq t_2^i - kc_2^i$ (comming from dominance in node i). Row two treats the case when $\max\{0, e_j - (t_1^i + t_{ij})\} = e_j - (t_1^i + t_{ij})$ and $\max\{0, e_j - (t_2^i + t_{ij})\} = 0$, where the later gives the second inequality in the line. In the third row $\max\{0, e_j - (t_1^i + t_{ij})\} = 0$ and $\max\{0, e_j - (t_2^i + t_{ij})\} = e_j - (t_2^i + t_{ij})$. The first inequality in the third row holds true as $e_j - (t_2^i + t_{ij}) \geq 0$. In the last row $\max\{0, e_j - (t_1^i + t_{ij})\} = e_j - (t_1^i + t_{ij})$ and $\max\{0, e_j - (t_2^i + t_{ij})\} = e_j - (t_2^i + t_{ij})$. The inequality in the fourth line is true as the dominance criteria $c_1^i \leq c_2^i$ hold in node i and $\varepsilon_2 \geq 0$. □

This concludes the proof of Claim 4. ■

In this section we have presented dominance criteria for the solution of subproblems A and B, and proven that by using these criteria the optimal route will never be lost (dominated). In Chapter 7 these dominance criteria will be tested and compared to the use of the weaker dominance criteria (D.1)–(D.3).

[4]Note that the cost of travelling the arc (i, j), and the service time at node j are both—for path 2—greater or equal to those for path 1, due to the battery difference between them in node i ($battery(\mathcal{L}_{i1}) \geq battery(\mathcal{L}_{i2})$).

6.3 Speeding up the solution process

While the solution process presented in this chapter can be used as is, some changes to it can be made to it in an attempt to improve the solution speed. In this section we present some suggestions of changes that could be used.

6.3.1 Solve a non-elementary shortest path problem

The ESPPRC is an NP-hard problem (in the strong sense) [40]; it is thus a problem that is highly time consuming to solve to optimality for large problem instances. The solution time could be reduced if non-elementary paths where allowed to be generated (i.e., paths containing cycles), which would turn the subproblem into a SPPRC (shortest path problem with resource constraints) for which there exist pseudo-polynomial time algorithms [41]. However, the lower bounds on the optimal objective value obtained by solving a SPPRC column generation subproblem will in general be weaker than if the subproblem is an ESPPRC; also the upper bound resulting from this approach will in general be weaker. One can tighten these bounds by implementing a compromise between the SPPRC and the ESPPRC, by forbidding the paths to have short cycles (a common approach is to exclude cycles including only two nodes).

6.3.2 Early termination of the column generation algorithm

Another approach to speed up the solution process is early termination. E.g. when the upper and lower bounds are very close, so that any further time spent on solving could only result in a small improvement. Since the column generation is applied to a continuous relaxation of a mixed integer optimization problem, there is no guarantee anyway that the resulting solution will be optimal to the original problem. An upper bound for the optimal objective value to the IMP is the best objective value for RMP with integer constraints (IRMP) that is found. Next, we derive a lower bound on the optimal value of the IRMP, in line with the presentation in [44].

Given the (RMP) program with its dual variables $\beta_i \geq 0$ and $\alpha \leq 0$, corresponding to its constraints, the inequality

$$z - \sum_{i \in V_c} \beta_i - F_{\max}\alpha \geq \sum_{r \in \hat{R}} \lambda_r (c_r - \sum_{i \in V_c} \gamma_{ir}\beta_i - \alpha)$$

holds. Letting $f_r := c_r - \sum_{i \in V_c} \gamma_{ir}\beta_i - \alpha$, $r \in R$, it follows that

$$z - \sum_{i \in V_c} \beta_i - F_{\max}\alpha \geq \sum_{r \in \hat{R}} \lambda_r f_r \geq \min_{r \in \mathcal{R}}\{f_r\} \sum_{r \in \hat{R}} \lambda_r \geq \min_{r \in \mathcal{R}}\{f_r\} F_{\max},$$

where the last inequality holds since $\min_{r \in \mathcal{R}} f_r \leq 0$. Finally we arrive at the lower bound on z, given by

$$z \geq \min_{r \in \mathcal{R}}\{f_r\} F_{\max} + \sum_{i \in V_c} \beta_i + F_{\max}\alpha.$$

Even though this allows us to terminate the column generation process after fewer iterations, it is time consuming to obtain these lower bounds as the subproblem must be solved to optimum to get the value of $\min_{r \in \mathcal{R}}\{f_r\}$. One way to reduce this time loss is to update the lower bound, not in every iteration but for example every 10th iteration, or by generating bounds on separate (detached) strings. Another idea is to solve the SPPRC to optimality instead, to get a (weaker) lower bound as these problems can be solved faster than the EPPRC (when the subproblem is large).

6.3.3 Other small tweaks

- Relax some of the dominance criteria in the subproblems in the beginning of the column generation. This will cause dominance to happen more often and fewer routes needed to be checked each time the subproblem is solved. In the end, however, the subproblem must be solved with the correct dominance criteria to guarantee an optimal solution, if one wants to derive lower bounds for the RMP.
- Delete non-used columns. When the column generation has been running for some iterations the number of generated routes (i.e., columns) may be so large that the RMP becomes impractical to solve. As many of these columns are never used in the solution of the RMP one could consider to remove some of them. This does not change the lower bound for the hybrid VRP, but it can make the upper bound weaker. Implementation of this must,however, be done carefully as this may create an infinite loop of inserting and removing certain columns in RMP.
- Warm-start the dynamic programming algorithm by the use of a heuristic to find good paths quickly. Doing this in the beginning of the procedure is likely to considerably decrease the number of labels created as it will yield dominance at an early stage.

7

Tests and Results

We examine the use of studying hybrid VRPs, as well as how hard the problems are to solve, and how effective the column generation procedure is for solving these problems. This is done by solving problems A and B for some test instances. We have used the test instances for electrical VRP with time windows presented in [23] and found here.

7.1 The test setup

The test instances are created such that all of them possess a feasible solution for routing of electric vehicles, and hence also for VRPs with hybrid vehicles with the same battery functionality. The instances solved are chosen at random from a collection of differently structured problems. The nodes for each instance are either generated in clusters (marked with a "c" in the beginning of the name), random in a uniform matter (marked with "r"), or a mixture of these (marked with "rc"). The number of customer nodes are denoted by C# in the end of the name, where # is the number of customers. Finally, a testing instance is either tightly constrained in terms of time windows (the time windows for the customer nodes are small) or not. The instances with small time windows are denoted by a three-digit number < 200 and the other with a three-digit number ≥ 200. For example, in the testing instance r204C15 all the nodes are uniformly random positioned, large time windows and comprises fifteen customer nodes.

To enable a reconstruction of the calculations, we have made the following assumptions for the parameters: We assume a diesel price of €1.35 per litre, an electricity price of €0.03 per kWh and a ten times higher electricity consumption rate (kWh/km) than fuel consumption rate (l/km).[1]

All problem instances where solved on an Ubuntu 14.04 machine, Intel® Xeon® $CPU\ E5-$ 2683 v3 @ 2.00GHz × 56, 132GB RAM. The modelling software AMPL 12.1.0 with the optimization solver CPLEX 12 [45, 39] has been used to solve the RMP as well as the flow formulations of Problems A and B. The dynamic programming algorithm for solving subproblems (Algorithm 2) has been implemented in C++. All communication between C++ and AMPL is done using AMPL API for

[1] The electricity and diesel prices where taken in June 2016, from www.elspot.nu and www.bensinpriser.se respectively. The idea that electricity consumption rate is 10 times higher than the fuel consumption rate ($r^{\mathrm{El}} = 10r^{\mathrm{Fu}}$) is based on approximate values used in an internal Volvo report ER-59278. Here, the relation between fuel and electricity consumption is given as $r^{\mathrm{El}} = r^{\mathrm{Fu}} \cdot Q_{\mathrm{diesel}}/(3.6 \cdot 10^6)$ where $Q_{\mathrm{diesel}} = 3.6 \cdot 10^7$J/l is the energy density of the diesel.

C++(beta version). CPLEX runs on multiple threads by default while the dynamic programming algorithm runs on a single thread.[2]

An upper time limit of ten seconds was used for the solution of subproblems, such that the solving of the subproblem was terminated if a route with negative reduced cost was found and the problem had been running for more than ten seconds.

7.2 Solving hybrid VRPs

Section 7.2.1 presents computational results for different the ways of solving Problem A and B, while in Section 7.2.3 the results from the hybrid VRPs are compared to solutions for electric as well as for conventional fueled vehicles. When solving the problems we let the maximum number of routes equal the number of customer nodes, i.e., $F_{\max} := |\mathcal{V}_{\text{cus}}|$.

7.2.1 Comparison of different solution methods

We compare the following methods for solving Problems A and B

- CG: The problems are solved using column generation, as described in Chapter 6, where the RMP is solved using CPLEX and the ESPPRC subproblem is solved using Algorithm 2, using the dominance criteria from Section 6.2.
- CG Basic: Column generation is used and the subproblems are solved with the basic dominance criteria (D.1)–(D.3) (used as reference to our stronger dominance criteria).
- FLOW: The flow formulation of problems A and B are solved directly using AMPL and CPLEX.
- CG Relaxed: The column generation procedure with the dominance criteria from Section 6.2, but where the battery and cargo criteria are relaxed in the beginning of the column generation process. The relaxed criteria are reinforced when no more column with a negative reduced cost can be found without these.
- CG 2-cycles: Column generation where the elementary requirement of the routes generated is relaxed, to allow for subcycles containing more than two nodes.

All usage of the CPLEX solver is done with the default CPLEX options. Solution times and objective values for Problem A are presented in Tables 7.1 and 7.2 for the different solution methods. The results for Problem B are analogously presented in Tables 7.3 and 7.4. A two hours limit on the computing time (not CPU time) has been used, such that any instance running longer than this limit is canceled. When the solution process has been canceled due to a too long execution time, the time is marked as "—"; if an objective value is given, it is simply the best objective value

[2]Running the dynamic programming algorithm on more than one thread would be a logical step if there where more than one kind of subproblem to solve.

found up to this time. The solution times in Table 7.1 (Problem A) are compared in Figure 7.1, while the solution times in Table 7.3 are plotted in Figure 7.2.

Table 7.1: Results from solving Problem A for a selection of problem instances. Upper and lower bounds on the optimal objective values received by the CG and the CG Basic methods and the corresponding solution times (CPU seconds). For the solution to the flow formulation the best objective value found is presented along with its corresponding solution time (CPU seconds).

	CG			CG Basic			Flow model	
Instance	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	t [CPU s]
c101C5	240.6	240.6	0.0482	240.6	240.6	0.0160	240.6	3.02
r202C5	128.8	128.8	0.104	128.8	128.8	0.132	128.8	4.56
rc108C5	264.2	264.2	0.0520	264.2	264.2	0.0523	264.2	57.1
c101C10	386.9	383.1	0.344	386.9	383.1	0.252	386.3	$2.13 \cdot 10^3$
c104C10	279.9	279.9	2.88	279.9	279.9	5.80	279.9	$5.40 \cdot 10^4$
r102C10	262.9	262.9	0.408	262.9	262.9	0.448	262.9	$6.17 \cdot 10^4$
r103C10	197.5	197.5	1.08	197.5	197.5	1.52	197.5	$1.61 \cdot 10^5$
rc102C10	427.0	427.0	0.140	427.0	427.0	0.176	427.0	$4.81 \cdot 10^4$
rc108C10	346.9	346.9	0.744	346.9	346.9	0.592	346.9	$3.58 \cdot 10^4$
c103C15	369.9	367.8	8.71	369.9	367.8	8.54	369.9	—
c106C15	289.2	289.2	1.10	289.2	289.2	1.63	289.2	$8.85 \cdot 10^4$
c202C15	384.3	384.3	10.1	384.3	384.3	21.3	384.3	—
c208C15	300.5	300.5	5.52	300.5	300.5	12.5	300.5	$1.12 \cdot 10^5$
r102C15	409.0	409.0	1.75	409.0	409.0	2.16	409.0	—
r105C15	338.3	336.2	1.86	338.3	336.2	1.32	338.3	—
r202C15	358.0	358.0	61.1	358.0	358.0	72.8	376.3	—
r209C15	273.1	273.1	37.0	273.1	273.1	83.2	273.1	—
rc103C15	393.9	393.9	5.50	393.9	393.9	4.40	393.9	—
rc108C15	378.4	378.4	4.74	378.4	378.4	9.36	381.9	—
rc202C15	396.9	396.9	6.87	396.9	396.9	17.3	396.9	—

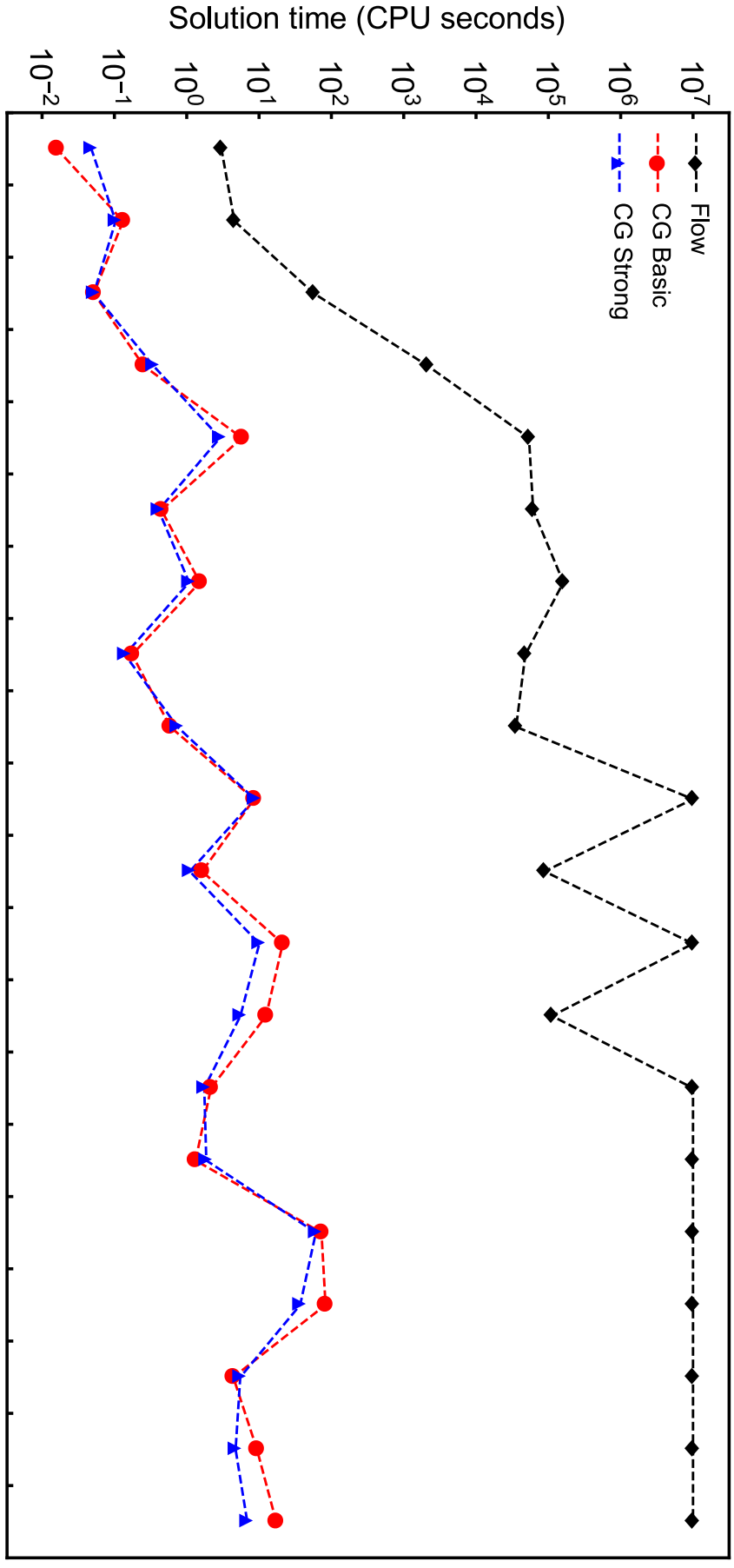

Figure 7.1: The solution times from Table 7.1 (Problem A). Problems that are not solved within the time limit of two hours are represented in the plots with a value of 10^7 CPU seconds.

Table 7.2: Bounds on the optimal objective values of instances (for Problem A) and corresponding CPU times for the CG Relaxed and CG 2-cycle methods.

	CG Relaxed			CG 2-cycle		
Instance	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	$\underline{z}$	t [CPU s]
c101C5	240.6	240.6	0.020	240.6	240.6	0.016
r202C5	128.8	128.8	0.092	157.0	124.1	5.66
rc108C5	264.2	264.2	0.048	297.5	231.4	0.064
c101C10	386.9	383.1	0.220	386.9	383.1	0.308
c104C10	279.9	279.9	4.42	322.6	255.4	3.08
r102C10	262.9	262.9	0.224	268.2	253.8	0.308
r103C10	197.5	197.5	0.752	197.5	197.3	1.73
rc102C10	427.0	427.0	0.152	427.0	427.0	0.171
rc108C10	346.9	346.9	0.396	450.6	303.7	0.576
c103C15	375.8	367.8	6.02	382.8	352.9	4.88
c106C15	289.2	289.2	0.556	289.2	283.7	0.996
c202C15	384.3	384.3	8.61	427.7	380.4	6.08
c208C15	300.5	300.5	3.68	349.4	251.0	32.3
r102C15	409.0	409.0	0.944	409.8	407.8	1.25
r105C15	338.3	336.2	0.448	338.3	336.2	1.00
r202C15	358.0	358.0	26.2	391.7	291.7	115
r209C15	273.1	273.1	18.3	301.9	249.7	$1.09 \cdot 10^3$
rc103C15	393.9	393.9	2.54	395.6	393.2	5.12
rc108C15	378.4	378.4	4.55	489.5	294.0	20.4
rc202C15	396.9	396.9	4.95	470.6	288.5	25.6

Table 7.3: Bounds on the optimal objective values and the corresponding CPU time, for the instances for Problem B, solved with the CG and CG Basic methods. For the FLOW model solution the best feasible objective value found is presented along with the solution time.

	CG			CG Basic			Flow model	
Instance	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	t [CPU s]
c101C5	240.6	240.6	0.048	240.6	240.6	0.212	240.6	3.02
r202C5	128.8	128.8	0.180	128.8	128.8	5.62	128.8	4.56
rc108C5	253.9	253.9	0.124	253.9	253.9	0.820	253.9	9.56
c101C10	386.9	383.1	1.72	386.9	383.1	$1.37 \cdot 10^2$	386.3	—
c104C10	273.9	273.9	14.7	—	—	—	273.9	$5.49 \cdot 10^4$
r102C10	249.2	249.2	1.20	249.2	249.2	28.8	249.2	—
r103C10	197.5	197.5	2.45	197.5	197.5	$2.82 \cdot 10^2$	197.5	—
rc102C10	422.3	422.3	0.308	422.3	422.3	9.78	422.3	—
rc108C10	345.9	345.9	1.03	345.9	345.9	67.2	345.9	$1.01 \cdot 10^5$
c103C15	370.3	360.0	67.9	—	—	—	361.2	—
c106C15	275.1	275.1	3.91	275.1	275.1	$3.60 \cdot 10^2$	275.1	$2.63 \cdot 10^4$
c202C15	376.5	371.4	37.2	—	—	—	371.5	—
c208C15	300.5	300.5	16.8	—	—	—	300.5	—
r102C15	409.0	408.4	8.49	409.0	408.4	$2.63 \cdot 10^4$	408.4	—
r105C15	336.2	329.4	3.71	339.7	329.4	$1.35 \cdot 10^4$	336.2	—
r202C15	358.0	358.0	145	—	—	—	358.0	—
r209C15	273.1	273.1	108	—	—	—	273.1	—
rc103C15	393.9	393.9	5.97	393.9	393.9	$7.16 \cdot 10^2$	393.9	—
rc108C15	364.2	364.2	23.2	—	—	—	364.2	—
rc202C15	396.9	396.9	22.6	—	—	—	396.9	—

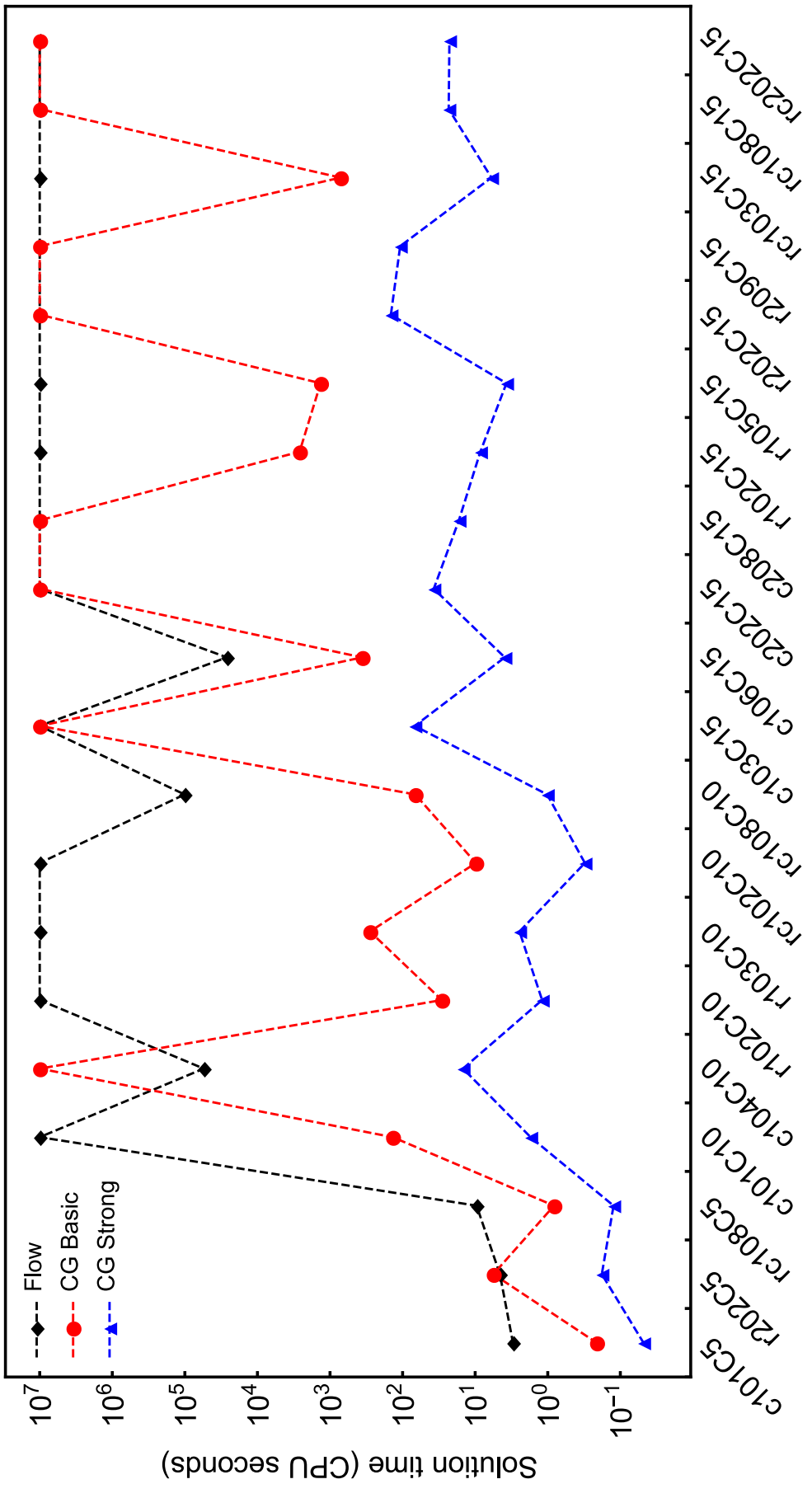

Figure 7.2: The solution times from Table 7.3 (Problem B). Problems that are not solved within two hours of computing time (non-CPU time) are represented by a value of 10^7 CPU seconds.

Table 7.4: Bounds on the optimal objective values and solution (CPU) times are presented here for Problem B solved using the CG Relaxed and CG 2-cycle methods.

	CG Relaxed			CG 2-cycle		
Instance	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	$\underline{z}$	t [CPU s]
c101C5	240.6	240.6	0.032	240.6	240.6	0.216
r202C5	128.8	128.8	0.056	161.9	119.4	34.5
rc108C5	253.9	253.9	0.068	308.2	206.1	0.156
c101C10	386.9	383.1	1.07	397.2	373.6	10.6
c104C10	273.9	273.9	5.54	318.9	239.6	15.7
r102C10	249.2	249.2	0.344	249.2	248.8	1.08
r103C10	197.5	197.5	0.556	197.5	197.3	3.50
rc102C10	422.3	422.3	0.164	422.3	422.3	0.316
rc108C10	345.9	345.9	0.396	419.2	292.2	1.71
c103C15	370.3	360.0	18.2	370.9	343.5	17.41
c106C15	275.1	275.1	1.24	275.1	275.1	5.40
c202C15	371.5	371.4	7.47	390.9	354.4	24.0
c208C15	300.5	300.5	5.08	—	—	—
r102C15	409.0	408.4	3.09	409.8	407.8	5.33
r105C15	338.3	329.4	0.98	360.3	302.9	3.60
r202C15	358.0	358.0	29.1	384.8	283.2	$5.04 \cdot 10^4$
r209C15	273.1	273.1	19.1	—	—	—
rc103C15	393.9	393.9	2.94	405.6	356.5	18.1
rc108C15	364.2	364.2	8.94	480.5	293.7	64.1
rc202C15	396.9	396.9	7.01	470.6	288.5	98.7

We also solved Problems A and B using CG Relaxed for the three instances given in Appendix B of slightly larger size than the instances regarded in Tables 7.1–7.4. The results are presented in Table 7.5 and shows further how solution times of Problem B are longer than for Problem A.

Table 7.5: Objective values and solution times (CPU time) for Problem A and B for the test instances presented in Appendix B, solved using the CG Relaxed method. The computations were terminated after exceeding ten hours of computing time.

		CG Relaxed (Problem A)			CG Relaxed (Problem B)		
# customers	# recharging nodes	$\overline{z}$	$\underline{z}$	t [CPU s]	$\overline{z}$	$\underline{z}$	t [CPU s]
20	4	269.24	269.24	224	269.24	269.24	553
26	7	438.49	401.27	780	—	—	—
30	7	362.67	326.16	18600	341.22	325.87	23200

The hybrid VRPs, Problem A and Problem B, are seemingly heavy to solve. For the flow formulations a few of the larger instances of Problem A and more than two thirds of all instances of Problem B were not solved within two hours of CPLEX computing. Column generation with the domination criteria from Section 6.2 (CG), on the other hand, solves each instance for both problems within two and

a half minute. The use of the stronger dominance criteria from Section 6.2 reduces the solution time for Problem B a lot compared to the basic criteria (D.1)–(D.3) (instances with solution times over two hours for CG Basic are solved within three minutes for CG; see Table 7.3). For Problem A, however, based on the data in Tables 7.1 and 7.2, there seems to be little difference between the use of the basic criteria and the tighter ones. We also see, from the data in Tables 7.1–7.4, that relaxation of some dominance criteria in the beginning of the column generation seems to reduce the solution time, while still in general reaching as small optimality gaps as with all criteria present.

The classical approach when solving VRPs using column generation is to solve a relaxation of the subproblem, that allows non-elementary routes to be generated. The reason for this is that pseudo polynomial algorithms exist for these. This means that the solution times for solving large instances of these kinds of problems is likely not to increase as much as when considering elementary shortest path problems. In this report, however, we have not implemented any of these pseudo-polynomial algorithms in CG 2-cycles procedure. Instead, CG 2-cycles is a generalization of the CG method, such that visited nodes are only made temporarily unreachable. The reason for considering CG 2-cycles is to find out how much larger the optimality gaps resulting from solving these (weaker) subproblems is likely to become. The optimality gaps resulting from CG are typically much smaller than those from CG 2-cycles; which is the reason that we have focused on ESPPTW as subproblems instead of SPPTW.

7.2.2 The effect of time window sizes

It is not only the number of nodes that affects the solution time. Figure 7.3 illustrates—for the CG relaxed method—the solution times as functions of the number of customer nodes, for Problems A and B, with large as well as tight time windows. As can be seen, the solution times for the column generation method is greatly dependent on the number of nodes in the problem, but also how tight the constraints are.

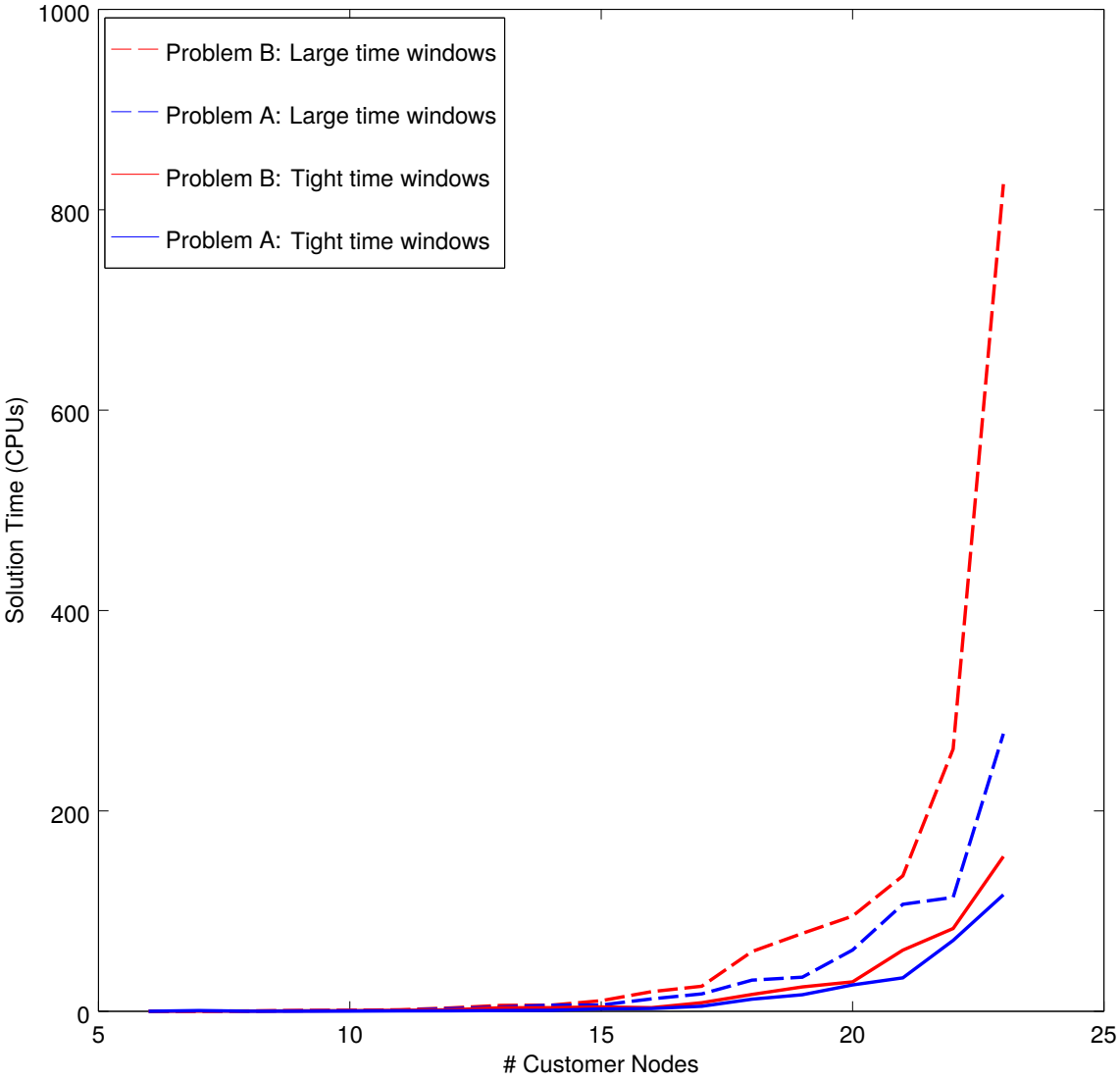

Figure 7.3: The solution time increases for Problems A and B when new customer nodes are added to the instance. The added nodes are randomly chosen from the set of customer nodes in the test instance r102_21. The dashed lines correspond to time windows being equal to the depot time window, while the solid lines correspond to the tighter time windows from r102_21.

7.2.3 Comparison of solutions to the hybrid VRP to solution for problems with differently powered vehicles

In order to answer the question "Would routing of electric or conventional fuel vehicles generate routes comparable to the routes for hybrid vehicles?", we compare solutions from VRPs with electric as well as diesel vehicles to the ones for hybrid vehicles. To enable this comparison we solve the testing instances as electric VRPs as well as conventional fuel VRPs and look at the cost of using these routes with hybrid vehicles.

Table 7.6 shows the relative increase (in %) of the cost when optimally routing (i.e., the best solution found) the hybrid vehicles as being either pure electric or pure diesel. The first entry in this table can be understood in the following way: The cost of routing a fleet of hybrid vehicles the same way as is optimal to route a fleet of electric vehicles, gives a 3.86% higher total cost than if the problem was solved as a hybrid VRP (for problems with fixed recharging time).

Table 7.6: Improvement when optimizing for hybrids compared to running hybrids on solutions optimized for electric or diesel vehicles, respectively. Entries where optimum is not confirmed for the Electricity or Diesel solutions are marked with a dagger (†), while a double dagger (‡) marks non-confirmed solutions for the hybrid problems.

	Problem A		Problem B	
Instance	Electricity [%]	Diesel [%]	Electricity [%]	Diesel [%]
c101C5	2.70	7.03	2.70†	7.03†
r202C5	0	28.92	0	28.92
rc108C5	16.65	13.59†	0	18.20†
c101C10	3.86	15.50	1.94‡	15.50‡
c104C10	0	17.22	0	19.79
r102C10	0	9.03	0	15.02
r103C10	2.73	16.01	2.73	16.01
rc102C10	2.13	11.39†	2.84	12.63
rc108C10	12.78	25.21	0	25.57†
c103C15	0.49‡	7.92‡	2.91‡	10.52‡
c106C15	5.98†	5.57	0	10.98†
c202C15	3.62	24.37†	0‡	28.65‡
c208C15	0	25.43	0	25.43
r102C15	2.69	6.82	2.84	6.98
r105C15	1.74‡	15.49‡	0‡	16.22‡
r202C15	0	32.37	0	39.13
r209C15	7.36†	30.91	7.36	30.91
rc103C15	0.96	9.79†	0.96	9.79
rc108C15	0	13.67	1.65	19.19
rc202C15	2.17†	31.73	2.17†	31.73†

We can see how the routes between hybrid, electric and diesel vehicles can vary by looking at the solutions of instance r102C15 for the different problems, see Figures 7.4–7.8.

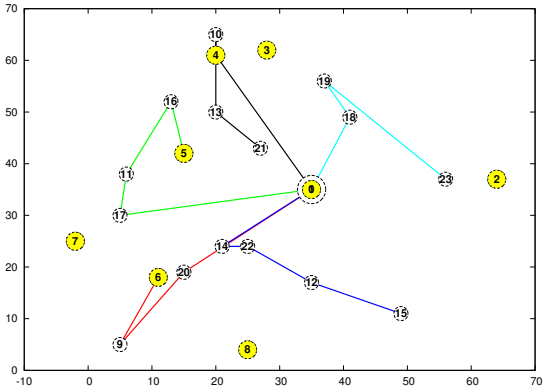

Figure 7.4: Best solution found to Problem A for the instance r102C15.

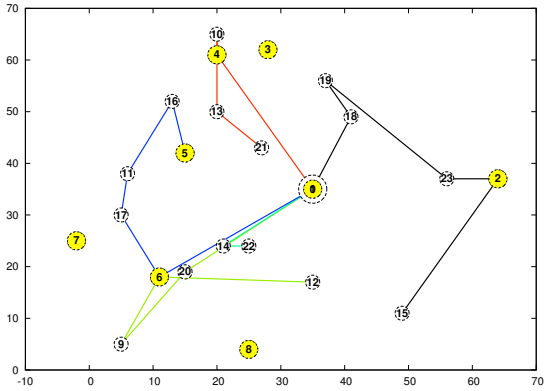

Figure 7.5: Best solution found for Problem A solved as an electric VRP for the instance r102C15.

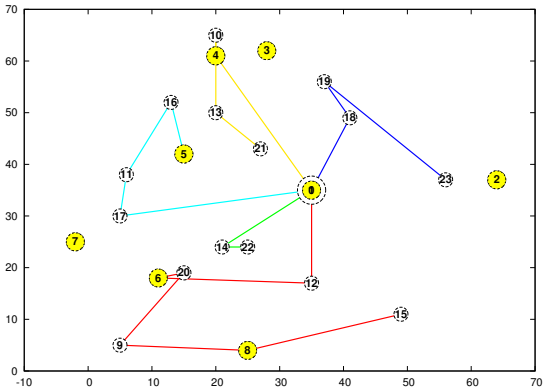

Figure 7.6: Best solution found to Problem B for the instance r102C15.

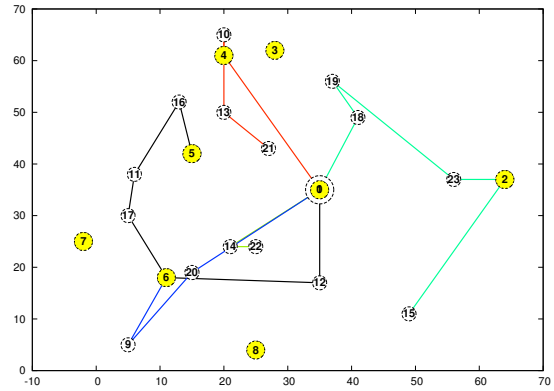

Figure 7.7: Best solution found for Problem B solved as an electric VRP for the instance r102C15.

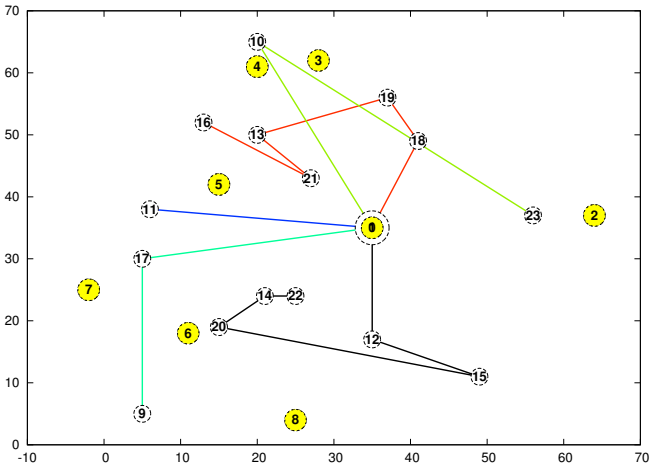

Figure 7.8: Best solution found for Problems A and B solved with diesel as the only propellant for the instance r102C15.

Table 7.6 reveals that the best routes found for electric vehicles often, but not always, are seemingly faire routes for hybrid vehicles too (this is, however, of course dependent on the scale of the problems). The electric VRPs seem to yield routes that are more useful for hybrid vehicles than the diesel VRPs do. The need for solving hybrid VRPs still remain though, not least as electric VRPs might not even have feasible solutions.

Lastly, electric vehicles are more dependent on recharging nodes than hybrid vehicles are. We investigated this by solving hybrid and electric VRP instances with varying numbers of recharging nodes. Figure 7.9 shows the best objective values found for an instance with fifteen customer nodes and 1–15 recharging stations. The positions of the recharging stations as well as of the customer nodes were chosen uniformly random. The blue dots represent the objective value for the best solution found for Problem B and the red line is the same for the electric VRP (Problem B without the diesel option).

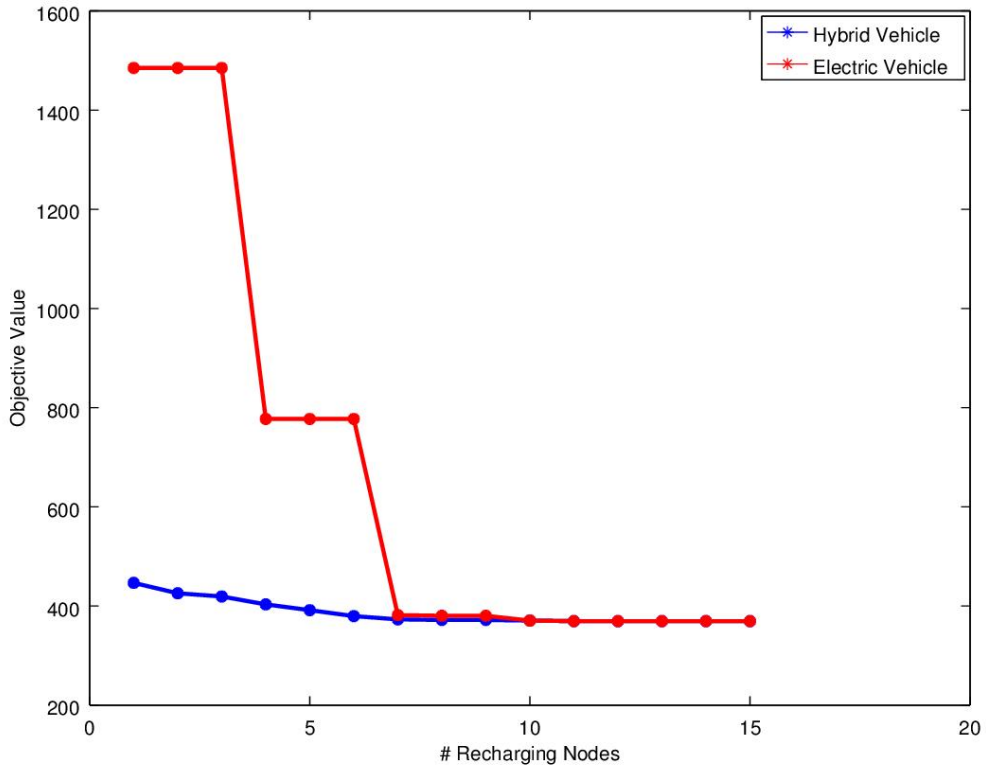

Figure 7.9: The costs of the best solutions found for a fleet of electric and corresponding hybrid vehicles, for different numbers of recharging nodes. For the full test instance, see Appendix B.

8

Conclusion and outlook

In this thesis we presented two VRPs with a homogeneous fleet of plug-in hybrid vehicles: Problem A, assuming fixed recharging times, and Problem B, in which the recharging times are assumed to depend on the battery charge level at arrival at the recharging station, the latter assumption being more realistic. The recharge of the vehicles' batteries can only take place at special recharging nodes, which are optional for the vehicles to visit. Refueling is not considered in the models as these type of stations are (generally) far more common compared to recharging stations, and the time for refueling is negligible compared to recharging times. To the best of the author's knowledge, neither of these models have been studied previously. The seemingly small difference between Problems A and B makes them differently hard to solve. The difference in complexity between Problems A and B is illustrated by the time required to solve their respective flow formulations. For Problem A, 55% of the instances tested (with up to 15 customer nodes, 5 recharging nodes and 15 vehicles) were solved within two hours of computing time, the corresponding figure for Problem B being 30%. With the flow formulations as our starting point we have formulated corresponding column generation models, which can be solved much faster.

The need for studying hybrid VRPs becomes evident when comparing their solutions with those of pure electric VRPs and pure conventionally fuelled VRPs. At a first glance, the electric VRP seems to serve as a good enough approximation of the hybrid VRP, in the sense that their respective optimal routes are similar. Indeed, for some instances this seems to be the case: when considering the recharging times being dependent on battery charge level, for only one out of our 20 test instances the best objective value found for the electric VRP is more than 3% higher than the corresponding value for the hybrid VRP. But, the electric VRP seems to be very sensitive to sparsity of the recharging stations, while the hybrid VRP does not. This suggests that routing hybrid vehicles as pure electric vehicles would be a bad idea, if the recharging nodes are scarce. Routing hybrid vehicles as conventional fuel vehicles seems to generate bad routes in general, which is natural since these routes will not visit any recharging nodes.

We solve the hybrid VRP using column generation, which divides the problem into a cardinality constrained set covering master problem and an elementary shortest path subproblem with resource constraints. The subproblems are solved using a labelling dynamic programming algorithm based on the one presented in [40], and for which our main contribution is to provide stronger (problem specific) dominance criteria which considerably improves the performance of the algorithm. The column generation algorithm yields tight bounds on the optimal value for all of the

test instances solved, and the solution times are generally much shorter than those emerging from solving the flow formulations directly using the CPLEX solver. While column generation using the basic dominance criteria from [40] in most cases solves the problems faster than the flow formulation solutions, it took over two hours to solve some of the instances with 10–15 customer nodes. The usage of the stronger dominance criteria greatly improves the solution speed, with the result that all instances were solved within three minutes.

Although the column generation method enables a faster solution of the hybrid VRPs than by solving their flow formulations directly using the default branch-and-cut method in CPLEX, we believe it to hold further improvement potential. Some changes that have not been implemented in this work, but that seem likely to improve the solution speed, are to "warm up" the subproblem solution by using a heuristic to create promising initial paths. To be able to tackle larger instances one could also terminate the column generation when the difference between the upper and lower bounds is below some threshold value. To generate lower bounds we need to solve the subproblems to optimality. In order not to slow down the solution process by solving all subproblems to optimality, one could generate these bounds on a separate detached thread, such that the column generation procedure does not have to wait for the new lower bound before it can continue on a new iteration.

A natural next step would be to apply the model on real data, such that the fully theoretical problem studied here is applied to routing of vehicles in real life. Other interesting areas would be to look at further extensions of the hybrid VRP, such as models with weight dependent fuel and electricity consumption rates or considering (time dependent) traffic congestion. Another interesting research topic is the optimization of the positioning of the recharging nodes, enabling a more efficient routing for both electric and hybrid vehicles.

Bibliography

[1] United Nations. Paris agreement — status of ratification. unfccc.int/paris_agreement/items/9444.php, 2016. Accessed: 2016-06-30.

[2] European Commission. Road transport: Reducing co2 emissions from vehicles. ec.europa.eu/clima/policies/transport/vehicles/index_en.htm, 2016. Accessed: 2016-06-27.

[3] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[4] E. Demir, T. Bektaş, and G. Laporte. A review of recent research on green road freight transportation. *European Journal of Operational Research*, 237:775–793, August 2014.

[5] Canhong Lin, K.L. Choy, G.T.S. Ho, S.H. Chung, and H.Y. Lam. Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, 41:1118 – 1138, 2014.

[6] T. Bektaş, E. Demir, and G. Laporte. Green vehicle routing. In Harilaos N. Psaraftis, editor, *Green Transportation Logistics*, volume 226 of *International Series in Operations Research & Management Science*, chapter 7, pages 243–265. Springer, 2016.

[7] İ. Kara, B.Y. Kara, and M.K. Yetis. Energy minimizing vehicle routing problem. In *Combinatorial Optimization and Applications: First International Conference, COCOA 2007, Xi'an, China, August 14-16, 2007. Proceedings*, volume 4616, pages 62–71. Springer Berlin Heidelberg, 2007.

[8] Ricardo Fukasawa, Qie He, and Yongjia Song. A branch-cut-and-price algorithm for the energy minimization vehicle routing problem. *Transportation Science*, 50(1):23–34, 2016.

[9] D. Chen, R.G. Batson, and Y. Dang. *Applied Integer Programming. Modeling and Solution*. Wiley, 1st edition, 2010.

[10] E. Demir, T. Bektaş, and G. Laporte. A comparative analysis of several vehicle emission models for road freight transportation. *Transportation Research Part D: Transport and Environment*, 16(5):347 – 357, 2011.

[11] T. Bektaş and G. Laporte. The pollution-routing problem. *Transportation Research Part B*, 45:1232–1250, 2011.

[12] E. Demir, T. Bektas, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

[13] Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte. The fleet size and mix pollution-routing problem. *Transportation Research Part B: Methodological*, 70:239 – 254, 2014.

[14] Y. Xiao, Q. Zhao, I. Kaku, and Y. Xu. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research*, 39(7):1419–1431, 2012.

[15] Y. Kuo. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, 59(1):157–165, 2010.

[16] E. Demir, T. Bektaş, and G. Laporte. The bi-objective pollution-routing problem. *European Journal of Operational Research*, 232(3):464–478, 2014.

[17] W. Maden, R. Eglese, and D. Black. Vehicle routing and scheduling with time-varying data: A case study. *Journal of the Operational Research Society*, 61(3):515–522, 2010.

[18] A. Franceschetti, D. Honhon, T.V. Woensel, T. Bektas, and G. Laporte. The time-dependent pollution-routing problem. *Transportation Research Part B*, 56:265–293, February 2013.

[19] Y. Xiao and A. Konak. Green vehicle routing problem with time-varying traffic congestion. In *INFORMS Computing Society Conference Richmond*, volume 14, pages 134–148. 2015.

[20] J. Qian and R. Eglese. Fuel emissions optimization in vehicle routing problems with time-varying speeds. *European Journal of Operational Research*, 248(3):840–848, 2016.

[21] S. Erdogan and E. Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E*, 48:100–114, 2012.

[22] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

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

[24] F. Marra, G.Y. Yang, C. Traeholt, E. Larsen, C.N. Rasmussen, and S. 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.

[25] D. Goeke and M. Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81 – 99, 2015.

[26] S. Mancini. The hybrid vehicle routing problem. Optimization online `optimization-online.org/DB_HTML/2015/07/4991.html`, 2015.

[27] K. Salimifard and R. Raeesi. A green routing problem: optimising CO_2 emissions and costs from a bi-fuel vehicle fleet. *International Journal of Advanced Operations Management*, 6(1):27–57, 2014.

[28] R.G. Conrad and M.A. Figliozzi. The recharging vehicle routing problem. In *Conference Proceedings 61st Annual Conference IIE*, volume 4, pages 2785–2792. 2011.

[29] T. Abdallah. The plug-in hybrid electric vehicle routing problem with time windows. Master's thesis, University of Waterloo, Ontario, Canada, 2013.

[30] P. Lebeau, C. De Cauwer, J. Van Mierlo, C. Macharis, W. Verbeke, and T. Coosemans. Conventional, hybrid, or electric vehicles: Which technology for an urban distribution centre? *Hindawi Publishing Corporation The Scientific World Journal*, 2015.

[31] O. Arslan, B. Yildiz, and O.E. Karaşan. Minimum cost path problem for plug-in hybrid electric vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 80:123 – 141, 2015.

[32] O. Arslan and O.E. Karaşan. A benders decomposition approach for the charging station location problem with plug-in hybrid electric vehicles. *Transportation Research Part B: Methodological*, 93, Part A:670 – 695, 2016.

[33] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[34] N. Andréasson, A. Evgrafov, and M. Patriksson. *An Introduction to Continuous Optimization: Foundations and Fundamental Algorithms*. Studentlitteratur AB, 2007.

[35] R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, chapter 9, pages 85–103. Springer US, Boston, MA, 1972.

[36] P. Gács and L. Lovász. Khachiyan's algorithm for linear programming. In H. König, B. Korte, and K. Ritter, editors, *Mathematical Programming at Oberwolfach*, volume 14, chapter 4, pages 61–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 1981.

[37] G.B. Dantzig and M.N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer, 1st edition, 2003.

[38] IBM ILOG CPLEX Optimization Studio. www-03.ibm.com/software/products/sv/ibmilogcpleoptistud, 2016. Accessed: 2016-07-22.

[39] IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.studio.help/pdf/usrcplex.pdf, 2015. Version 12, Release 6.

[40] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[41] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[42] E. Danna and C. Le Pape. *Branch-and-Price Heuristics: A Case Study on the Vehicle Routing Problem with Time Windows*, pages 99–129. Springer US, Boston, MA, 2005.

[43] B. Kallehauge, J. Larsen, O.B.G. Madsen, and M.M. Solomon. Vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 3, pages 67–98. Springer US, Boston, MA, 2005.

[44] L.S. Lasdon. *Optimization Theory for Large Systems*. Dover books on Mathematics. Dover Publications, 1970.

[45] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Scientific Press series. Thomson, 2nd edition, 2003.

Appendices

A

Appendix A

Detailed derivation of master and subproblem

The master and subproblem for column generation for Problems A and B presented in this report can be derived through the theory presented on column generation and Dantzig-Wolfe decomposition in Chapter 4.3. We show this here for Problem A, Problem B can be treated similarly.

Problem A can be expressed as to

$$\underset{x,\tau,u,y,z}{\text{minimize}} \quad \sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}}(c^{\text{Fu}}z_{ij}^{\text{Fu},k}+c^{\text{El}}z_{ij}^{\text{El},k}), \tag{A1}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{K}}\sum_{j:(i,j)\in\mathcal{A}}x_{ij}^{k}=1, \qquad i\in\mathcal{V}_{\text{cus}}, \tag{A2}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}_{\text{rec}}\cup\mathcal{V}_{\text{cus}}}x_{0j}^{k}\leq F_{\max}, \tag{A5}$$

$$x_{ij}^{k}\in\{0,1\}, \qquad (i,j)\in\mathcal{A},k\in\mathcal{K} \tag{A14}$$

subject to the additional constraints (A3)–(A4),(A6)–(A13),(A15)–(A16).

The set described by the additional constraints is not a convex set, however every point in this set can still be written as a convex combination of its extreme points of the convex hull (i.e., the smallest convex set that includes all points in the additional constraint set). Denote these extreme points as u_l, $l\in\mathcal{L}$, and note that each of these extreme points represents a schedule specifying a route for each vehicle $k\in\mathcal{K}$ (such that not all costumers has to be visited). We then end up with the master problem to

$$\underset{\lambda}{\text{minimize}} \quad \sum_{l\in\mathcal{L}}c(u_l)\lambda_l, \tag{$A_1$1}$$

$$\text{subject to} \quad \sum_{l\in\mathcal{L}}\chi_i(u_l)\lambda_l=1, \qquad i\in\mathcal{V}_c, \tag{$A_1$2}$$

$$\sum_{l\in\mathcal{L}}\mu(u_l)\lambda_l\leq F_{\max}, \tag{$A_1$3}$$

$$\sum_{l\in\mathcal{L}}\lambda_l=1, \tag{$A_1$4}$$

$$\lambda_l\in 0, \qquad l\in\mathcal{L}, \tag{$A_1$5}$$

where Ω_u denotes the set of all schedules (extreme points) u_l, $c(u_l)$ denotes the cost of schedule u_l, $\chi_i(u_l)$ denotes the number of times node i is visited for schedule

u_l, and $\mu(u_l)$ denotes the number of vehicles used for schedule u_l. But there is no real need to regard columns (i.e., variables u_l) that represent schedules for the whole fleet of vehicles; instead we will consider separate schedules for each vehicle. Therefore, by letting v_r represent the route of a vehicle, and letting $\mathcal{R} := \{r : v_r \text{ is a (by the additional constraints) feasible route }\}$ be the index set of all possible routes, our master problem can be expressed as to

$$\begin{aligned}
\underset{\lambda}{\text{minimize}} \quad & \sum_{r\in\mathcal{R}} c(v_r)\lambda_r, && && \text{(G1)}\\
\text{subject to} \quad & \sum_{r\in\mathcal{R}} \chi_i(v_r)\lambda_r = 1, && \forall i \in \mathcal{V}_c, && \text{(G2)}\\
& \sum_{r\in\mathcal{R}} \lambda_r \leq F_{\max}, && && \text{(G3)}\\
& 0 \leq \sum_{r\in\mathcal{R}} \lambda_r \leq F_{\max}, && && \text{(G4)}\\
& \lambda_r \geq 0, && r \in \mathcal{R}, && \text{(G5)}
\end{aligned}$$

where (G4) is true since a schedule (u_l) for all vehicles contain up to $F_{\max}$ vehicles, the other relations being straightforward to check. Obviously, the constraint (G4) is redundant, and we can thus rewrite the problem (G) as to

$$\begin{aligned}
\underset{\lambda}{\text{minimize}} \quad & \sum_{r\in\mathcal{R}} c(v_r)\lambda_r, && && \text{(G1)}\\
\text{subject to} \quad & \sum_{r\in\mathcal{R}} \chi_i(v_r)\lambda_r = 1, && \forall i \in V_c, && \text{(G2)}\\
& \sum_{r\in\mathcal{R}} \lambda_r \leq F_{\max}, && && \text{(G3)}\\
& \lambda_r \geq 0, && r \in \mathcal{R}. && \text{(G5)}
\end{aligned}$$

Finally, as the VRP instance solved are defined such that it is never cheaper to pass through a customer node than to skip it, we may, without loss of generality, allow more than one visit to each customer node. Then by letting the number of visits to a node i by route v_r be written as γ_{ir}, we end up with the same master problem as (IMP), namely to

$$\begin{aligned}
\underset{\lambda}{\text{minimize}} \quad & \sum_{r\in\mathcal{R}} c(v_r)\lambda_r, && && \text{(G1)}\\
\text{subject to} \quad & \sum_{r\in\mathcal{R}} \gamma_{ir}(v_r)\lambda_r \geq 1, && \forall i \in \mathcal{V}_c, && \text{(G2)}\\
& \sum_{r\in\mathcal{R}} \lambda_r \leq F_{\max}, && && \text{(G3)}\\
& \lambda_r \geq 0, && r \in \mathcal{R}. && \text{(G5)}
\end{aligned}$$

With the objective of finding the route that is most violated in the latest solution

of the dual to the MP we receive the subproblem to

$$\begin{aligned} &\underset{x,y,u,\tau,z}{\text{minimize}} && \sum_{(i,j)\in A} (c^F z_{ij}^F + c^E z_{ij}^E - \beta_i) - \alpha \\ &\text{subject to} \end{aligned}$$

the constraints (A3)–(A4),(A6)–(A13), (A15)–(A16).

The cost of the propellant used to cover the route is $\sum_{(i,j)\in A} (c^F z_{ij}^F + c^E z_{ij}^E)$, while α and β_i are linear programming dual variables to the master problem, corresponding to constraints (G3) and (G2), respectively.

B

Appendix B

Three larger problem instances made by random selection of nodes from problem instance r102_21 (by M. Schneider, A. Stenger, and D. Goeke.).

20 customer nodes, total 25 nodes:

	xcoord	ycoord	demand	ReadyTime	DueDate	ServiceTime
0	40.0	50.0	0.0	0.0	1236.0	0.0
1	40.0	50.0	0.0	0.0	1236.0	0.0
2	73.0	52.0	0.0	0.0	1236.0	0.0
3	90.0	55.0	0.0	0.0	1236.0	0.0
4	55.0	79.0	0.0	0.0	1236.0	0.0
5	15.0	80.0	10.0	455.0	513.0	90.0
6	30.0	50.0	10.0	0.0	1136.0	90.0
7	30.0	52.0	20.0	0.0	1135.0	90.0
8	28.0	52.0	20.0	0.0	1133.0	90.0
9	28.0	55.0	10.0	13.0	70.0	90.0
10	25.0	50.0	10.0	0.0	1131.0	90.0
11	25.0	52.0	40.0	158.0	232.0	90.0
12	25.0	55.0	10.0	672.0	744.0	90.0
13	23.0	52.0	10.0	263.0	311.0	90.0
14	23.0	55.0	20.0	438.0	510.0	90.0
15	20.0	50.0	10.0	0.0	1126.0	90.0
16	20.0	55.0	10.0	0.0	1125.0	90.0
17	10.0	35.0	20.0	0.0	1112.0	90.0
18	10.0	40.0	30.0	0.0	1114.0	90.0
19	50.0	30.0	10.0	980.0	1064.0	90.0
20	50.0	35.0	20.0	0.0	1127.0	90.0
21	50.0	40.0	50.0	0.0	1131.0	90.0
22	48.0	30.0	10.0	0.0	1124.0	90.0
23	72.0	55.0	10.0	0.0	1113.0	90.0
24	70.0	58.0	20.0	0.0	1114.0	90.0

```
 m             := 4;     # The number of recharging nodes
 maxCharge     := 79.69;
 LoadCapacity  := 200.0;
 rateElConsump := 1.0;
 RechargeRate  := 3.39;
 AverageSpeed  := 1.0;
```

26 customer nodes, total 34 nodes:

```
        xcoord    ycoord     demand     ReadyTime  DueDate    ServiceTime     :=
 0      40.0       50.0       0.0        0.0        1236.0     0.0
 1      40.0       50.0       0.0        0.0        1236.0     0.0
 2      73.0       52.0       0.0        0.0        1236.0     0.0
 3      13.0       31.0       0.0        0.0        1236.0     0.0
 4      59.0       17.0       0.0        0.0        1236.0     0.0
 5      73.0       23.0       0.0        0.0        1236.0     0.0
 6      74.0       32.0       0.0        0.0        1236.0     0.0
 7      90.0       44.0       0.0        0.0        1236.0     0.0
 8      45.0       68.0       10.0       0.0        1127.0     90.0
 9      42.0       65.0       10.0       950.0      1026.0     90.0
 10     40.0       69.0       20.0       683.0      741.0      90.0
 11     40.0       66.0       20.0       0.0        1130.0     90.0
 12     50.0       40.0       50.0       0.0        1131.0     90.0
 13     48.0       30.0       10.0       0.0        1124.0     90.0
 14     48.0       40.0       10.0       67.0       139.0      90.0
 15     47.0       35.0       10.0       1068.0     1129.0     90.0
 16     47.0       40.0       10.0       0.0        1133.0     90.0
 17     45.0       30.0       10.0       0.0        1125.0     90.0
 18     45.0       35.0       10.0       0.0        1130.0     90.0
 19     95.0       30.0       30.0       390.0      444.0      90.0
 20     95.0       35.0       20.0       481.0      543.0      90.0
 21     53.0       30.0       10.0       904.0      954.0      90.0
 22     92.0       30.0       10.0       287.0      361.0      90.0
 23     53.0       35.0       50.0       78.0       142.0      90.0
 24     45.0       65.0       20.0       0.0        1130.0     90.0
 25     90.0       35.0       10.0       575.0      639.0      90.0
 26     88.0       30.0       10.0       534.0      600.0      90.0
 27     88.0       35.0       20.0       0.0        1095.0     90.0
 28     87.0       30.0       10.0       0.0        1094.0     90.0
 29     85.0       25.0       10.0       637.0      689.0      90.0
 30     60.0       60.0       10.0       0.0        1123.0     90.0
 31     67.0       85.0       20.0       459.0      519.0      90.0
 32     55.0       80.0       10.0       836.0      896.0      90.0
 33     55.0       85.0       20.0       744.0      798.0      90.0
;

param m := 7; # The number of recharging nodes
param maxCharge := 79.69;
param LoadCapacity := 200.0;
param rateElConsump := 1.0;
param RechareRate := 3.39;
param AverageSpeed := 1.0;
```

30 customer nodes, total 38 nodes

	xcoord	ycoord	demand	ReadyTime	DueDate	ServiceTime
0	40.0	50.0	0.0	0.0	1236.0	0.0
1	40.0	50.0	0.0	0.0	1236.0	0.0
2	39.0	96.0	0.0	0.0	1236.0	0.0
3	23.0	80.0	0.0	0.0	1236.0	0.0
4	28.0	13.0	0.0	0.0	1236.0	0.0
5	39.0	31.0	0.0	0.0	1236.0	0.0
6	48.0	11.0	0.0	0.0	1236.0	0.0
7	59.0	17.0	0.0	0.0	1236.0	0.0
8	45.0	68.0	10.0	0.0	1127.0	90.0
9	45.0	70.0	30.0	499.0	553.0	90.0
10	42.0	66.0	10.0	0.0	1129.0	90.0
11	42.0	68.0	10.0	584.0	656.0	90.0
12	42.0	65.0	10.0	950.0	1026.0	90.0
13	40.0	69.0	20.0	683.0	741.0	90.0
14	40.0	66.0	20.0	0.0	1130.0	90.0
15	38.0	68.0	20.0	1057.0	1125.0	90.0
16	38.0	70.0	10.0	958.0	1040.0	90.0
17	35.0	66.0	10.0	779.0	845.0	90.0
18	35.0	69.0	10.0	869.0	941.0	90.0
19	25.0	85.0	20.0	176.0	228.0	90.0
20	20.0	50.0	10.0	0.0	1126.0	90.0
21	20.0	55.0	10.0	0.0	1125.0	90.0
22	10.0	35.0	20.0	0.0	1112.0	90.0
23	10.0	40.0	30.0	0.0	1114.0	90.0
24	8.0	40.0	40.0	355.0	437.0	90.0
25	8.0	45.0	20.0	0.0	1113.0	90.0
26	5.0	35.0	10.0	655.0	725.0	90.0
27	5.0	45.0	10.0	0.0	1110.0	90.0
28	2.0	40.0	20.0	569.0	619.0	90.0
29	0.0	40.0	30.0	472.0	532.0	90.0
30	0.0	45.0	20.0	0.0	1105.0	90.0
31	35.0	30.0	10.0	143.0	199.0	90.0
32	35.0	32.0	10.0	0.0	1127.0	90.0
33	33.0	32.0	20.0	0.0	1126.0	90.0
34	33.0	35.0	10.0	789.0	849.0	90.0
35	32.0	30.0	10.0	0.0	1124.0	90.0
36	30.0	30.0	10.0	0.0	1123.0	90.0
37	30.0	35.0	10.0	698.0	754.0	90.0

```
m := 7; # The number of recharging nodes
maxCharge       := 79.69;
LoadCapacity    := 200.0;
rateElConsump   := 1.0;
RechargeRate    := 3.39;
AverageSpeed    := 1.0;
```

Instance with large number of recharging nodes

The full instances used for the test with different number of recharging nodes.

	xcoord	ycoord	demand	ReadyTime	DueDate	ServiceTime
0	35.0	35.0	0.0	0.0	230.0	0.0
1	35.0	35.0	0.0	0.0	230.0	0.0
2	28.0	62.0	0.0	0.0	230.0	0.0
3	51.0	7.0	0.0	0.0	230.0	0.0
4	63.0	12.0	0.0	0.0	230.0	0.0
5	33.0	49.0	0.0	0.0	230.0	0.0
6	66.0	4.0	0.0	0.0	230.0	0.0
7	16.0	31.0	0.0	0.0	230.0	0.0
8	0.0	7.0	0.0	0.0	230.0	0.0
9	54.0	54.0	0.0	0.0	230.0	0.0
10	55.0	5.0	0.0	0.0	230.0	0.0
11	59.0	69.0	0.0	0.0	230.0	0.0
12	27.0	8.0	0.0	0.0	230.0	0.0
13	19.0	45.0	0.0	0.0	230.0	0.0
14	12.0	27.0	0.0	0.0	230.0	0.0
15	44.0	9.0	0.0	0.0	230.0	0.0
16	31.0	52.0	27.0	25.0	35.0	10.0
17	55.0	5.0	29.0	97.0	107.0	10.0
18	67.0	5.0	25.0	0.0	176.0	10.0
19	17.0	34.0	3.0	0.0	201.0	10.0
20	26.0	52.0	9.0	166.0	176.0	10.0
21	53.0	43.0	14.0	150.0	160.0	10.0
22	45.0	65.0	9.0	77.0	87.0	10.0
23	20.0	26.0	9.0	138.0	148.0	10.0
24	50.0	35.0	19.0	177.0	187.0	10.0
25	45.0	20.0	11.0	0.0	201.0	10.0

```
m := 15; # The number of recharging nodes
maxCharge := 60.63;
LoadCapacity := 200.0;
rateElConsump := 1.0;
RechareRate := 0.49;
AverageSpeed := 1.0;
```