COLUMN GENERATION-BASED METHODS FOR THE ELECTRIC VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS

by
ECE NAZ DUMAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Doctor of Philosophy

Sabancı University
May 2022

ABSTRACT

COLUMN GENERATION-BASED METHODS FOR THE ELECTRIC VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS

ECE NAZ DUMAN

Industrial Engineering Ph.D DISSERTATION, May 2022

Dissertation Supervisor: Prof. Bülent Çatay

Dissertation Co-Supervisor: Asst. Prof. Duygu Taş Küten

Keywords: Electric Vehicles, Vehicle Routing, Time Windows, Column Generation, Flexible Delivery

The Electric Vehicle Routing Problem with Time Windows (EVRPTW) is an extension of the well-known Vehicle Routing Problem with Time Windows (VRPTW), where a fleet of electric vehicles (EVs) is used instead of conventional vehicles. EVs have a limited driving range, and thus, they may need to visit a station on the route to recharge their batteries. The duration spent to recharge the battery is linearly proportional to the amount of energy transferred. In this thesis, the EVRPTW and the EVRP with Flexible Delivery (EVRP-FD) are studied. The first problem is based on the classical VRPTW and assumes that for each customer only one delivery location and its related time window are provided. In the second problem, this assumption is generalized such that customers are offered the flexibility to specify multiple delivery locations for service to take place within different predetermined time windows. We develop exact and heuristic algorithms based on the Column Generation (CG) method that is embedded in Branch-and-Price (BP) and Branch-and-Price-and-Cut (BPC) procedures to obtain solutions for these problems. Pricing subproblems of those methods are solved by using different labeling algorithms all based on the Pulse algorithm or the *ng-route* algorithm and improved with the state-of-the-art

acceleration techniques including (i) a bidirectional search mechanism in which both forward and backward labels are created, (ii) a method to prevent solving the pricing sub-problem at each iteration, (iii) a procedure for dynamically eliminating arcs that connect customers to remote stations from the network during the path generation, (iv) a bounding procedure providing early elimination of sub-optimal routes, and (v) an integer programming model that generates upper bounds. The BPC is strengthened by employing a well-known set of valid inequalities. The methods proposed for solving the EVRPTW are evaluated by using a benchmark data set that includes instances with up to 100 customers and 21 stations and several new solutions are introduced while some existing solutions are improved. The BP procedure with the Pulse algorithm also provides a number of new solutions for the instances with 100 customers to the literature. For the EVRP-FD, we present a new data set including instances with up to 120 customers and an extensive computational study is performed to evaluate the performance of the BP algorithm applied with the bidirectional Pulse procedure.

ÖZET

ZAMAN PENCERELİ ELEKTRİKLİ ARAÇ ROTALAMA PROBLEMİ İÇİN SÜTUN TÜRETME ALGORİTMASINA DAYALI ÇÖZÜM YÖNTEMLERİ

ECE NAZ DUMAN

Endüstri Mühendisliği DOKTORA TEZİ, Mayıs 2022

Tez Danışmanı: Prof. Dr. Bülent Çatay

Tez Eş-Danışmanı: Dr. Öğr. Üyesi Duygu Taş Küten

Anahtar Kelimeler: Elektrikli Araçlar, Araç Rotalama, Zaman Pencereleri, Sütun Türetme Metodu, Esnek Teslimatlar

Zaman Pencereli Elektrikli Araç Rotalama Problemi (ZEARP), geleneksel araçlar yerine bir elektrikli araç (EA) filosunun kullanıldığı ve iyi bilinen Zaman Pencereli Araç Rotalama Problemi'nin (ZARP) bir uzantısıdır. EA'ların sınırlı bir sürüş menzili vardır ve bu nedenle bataryalarını şarj etmek için rotadaki bir istasyonu ziyaret etmeleri gerekebilir. Şarj için harcanan süre, aktarılan enerji miktarı ile doğrusal orantılıdır. Bu tezde ZEARP ve Esnek Teslimatlı Elektrikli Araç Rotalama Problemleri (ET-EARP) incelenmiştir. İlk problem klasik ZARP'ye dayanmaktadır ve problemde her müşteri için sadece bir teslimat yeri ve bununla ilgili zaman penceresinin sağlandığı varsayılır. İkinci problemde, bu varsayım, müşterilere birden fazla teslimat yeri ve her teslimat noktası için farklı bir zaman penceresi belirtme esnekliğinin sağlandığı şekilde genelleştirilmiştir. Bu problemlere çözüm elde etmek için kesin ve sezgisel algoritmalar geliştiririz. Bu algoritmalar Sütun Türetme yöntemine dayalı Dal-ve-Ücret (DÜ) ve Dal-ve-Ücret-ve-Kesit (DÜK) prosedürlerinden oluşmaktadır. Bu yöntemlerin ücretlendirme alt problemleri Pulse algoritmasına veya *ng-rota* algoritmasına dayanan farklı etiketleme algoritmaları kullanılarak çözülmektedir. Sütun türetme algoritmalarını iyileştirmek için kullanılan hızlandırma teknikleri şu şek-

ildedir: (i) hem ileri hem de geriye doğru etiketlerin oluşturulduğu iki yönlü bir arama mekanizması, (ii) her yinelemede ücretlendirme alt probleminin çözülmesini önleyen bir yöntem, (iii) rota oluşturma sırasında müşterileri ağdan uzak istasyonlara bağlayan bağlantıları dinamik olarak ortadan kaldırmak için bir prosedür, (iv) alt optimal yolların erken ortadan kaldırılmasını sağlayan bir sınırlama prosedürü ve (v) üst sınırlar oluşturan bir tamsayılı programlama modeli. DÜK iyi bilinen bir dizi geçerli eşitsizlik uygulanarak güçlendirilir. ZEARP için önerilen yöntemler 100'e kadar müşteri ve 21 istasyona sahip örnekleri içeren bir veri seti kullanılarak değerlendirmiş ve mevcut bazı çözümler iyileştirilirken birçok yeni çözüm de sunulmuştur. Ayrıca, Pulse algoritması ile DÜ yöntemi kullanılarak 100 müşterili bir veri seti daha çözdülmüş ve elde edilen birkaç çözüm iyileştirilmiştir. ET-EARP için ise 120'ye kadar müşteriye sahip örnekleri içeren yeni bir veri seti oluşturulmuştur ve bu örnekler üzerinde çift yönlü Pulse prosedürü ile uygulanan DÜ algoritmasının performansını değerlendiren kapsamlı bir hesaplama çalışması sunulmaktadır.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my primary thesis advisor Bülent Çatay and my co-advisor Duygu Taş Küten who guided me throughout this project and was very supportive of me at each step.

I wish to extend my special thanks to my thesis committee members Professors Tonguç Ünlüyurt, Semra Ağralı, Gizem Özbaygın, and Sina Rastani for their advice, time, and valuable comments throughout my study.

I would like to thank Mir Ehsan Sadati, Nozir Shokirov, Sina Rastani, and Amin Ahmadi Digehsara for the special help they provided and their patience with me.

I wish to express my special thanks to my family who has been always there for me.

I am very thankful to Professors Gündüz Ulusoy, Kemal Kılıç, Tonguç Ünlüyurt, Nilay Noyan, Ilker Birbil, Güvenç Sahin, Barıs Balcıoglu, Murat Kaya, Esra Koca, and Ali Rana Atılgan, for everything they have taught to me.

This study was funded by The Scientific and Technical Research Council of Turkey through Grant #118M412.

I wish to express my appreciation to the administrative staff of Sabancı University for the moral and material support they provide for me throughout my study and for all others who helped me during this process.

To my sister, mother and father

LIST OF ABBREVIATIONS

ALNS	Adaptive Large Neighborhood Search
AN	Augmented Node
EV	Electric Vehicles
BEV	Battery EVs
BP	Branch-and-Price
BPC	Branch-and-Price-and-Cut
CG	Column Generation
CSP	Constrained Shortest Path
ESPPRC	Elementary Shortest Path Problem with Resource Constraints
EVRP	Electric Vehicle Routing Problem
EVRPTW	Electric Vehicle Routing Problem with Time Windows
EVRP-FD	EVRP with Flexible Delivery Options
GVRP	Green Vehicle Routing Problem
GHG	Greenhouse Gas
HLA	Heuristic Labeling Algorithm
HCG	Heuristic Column Generator
ICEV	Internal Combustion Engine Vehicles
ICP	Intermediate Column Pool
IMP	Integer Master Problem
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
MP	Multiple-Partial
RLPMP	Restricted Linear Programming Master Problem
SP	Single-Partial
SoC	State of Charge
TPA	Two-Phased Algorithm
TS	Tabu Search
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem

VRPTW Vehicle Routing Problem with Time Windows

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The transport sector causes about 25% of the total greenhouse gas (GHG) emissions in the world (Muller, 2022). Road transport is the main contributor with a share of approximately 75% (IEA, 2019). The environmental issues related to fossil-fuel-dependent transportation have attracted a lot of attention in the last decade, and growing concerns have been raised in modern societies (Zhu and Hu, 2019). Consequently, governments have started setting ambitious targets to combat GHG emissions. The European Commission (EC) targets a 90% reduction in emissions by 2050 by making all transport modes more sustainable (EC, 2020). Urban transport is particularly important because road vehicles are mostly used in densely populated areas, which causes the concentration of emissions in the cities. Many European countries including Norway, Denmark, the Netherlands, Germany, and the UK announced plans to ban the sale of internal combustion engine vehicles (ICEVs) shortly and phase them out through 2050 (IEA, 2020).

A promising approach to meet the emission reduction targets is replacing the ICEVs with electric vehicles (EVs). Despite their high acquisition costs, EVs can be cost-effective in the long term due to their lower energy consumption per distance traveled (Wu et al., 2015). On the other hand, their range anxiety and limited recharging facility infrastructure constitute the main disadvantages (Giordano et al., 2018). Although managing a fleet of EVs is similar to managing an ICEV fleet, the operational limitations give rise to additional complexities for logistics service providers and call for new optimization approaches in operational planning. Therefore, studies that consider the utilization of EVs in logistics operations have grown rapidly in parallel with the advancements in the sector (Sassi and Oulamara, 2017), and many articles have appeared in the field of the Vehicle Routing Problem (VRP) during the past decade (see Qin et al., 2021 for an extensive review).

A growing interest in logistics industry has been concentrated parallel to the sustainability issues (Mangiaracina et al., 2015). In the last 30 years, the internet has been ubiquitously availed, which equalized the retail industry and enabled shopping

without geographic restrictions. According to the information reported by Digital Commerce (2021), in 2010 e-commerce was around 7% of the total retail purchases. This percentage has been steadily increasing until the surge caused by the Covid-19 pandemic. The worldwide online sales were around 15% of the total sales for the first three months of 2020. Around this time the Covid-19 pandemic started to spread through out the world and force people to stay at home. Therefore, the share of online sales increased by 58% in April 2021 (eMarketer, 2021). Online sales provide many advantages to the retailers, including reducing the costs related to the physical facilities and reaching distant customers. On the other hand, it also brings serious challenges related to the delivery of the goods to the end customers in the supply chain. The importance of the delivery to the end customers, last-mile logistics, is propelled to the forefront in developments of the logistics sector. A recent study reported by DHL (2021) points out that flexible delivery options can be offered for last-mile logistics. Flexible deliveries enable customers to provide different locations for the delivery to take place at different time windows.

Although EVs are invented in the mid-19th century, for many years they were not as commonly used as ICEVs since their driving ranges were quite limited (Guarnieri, 2012). Over the years the popularity of EVs is accelerated due to the following reasons: (i) technological developments, (ii) an increased focus on environmental issues and renewable energy, (iii) increasing public interest and awareness, (iv) the government incentives for the EV ownership, and (v) the structural incentives such as more and more stations being built in the urban areas (Zivin et al., 2014).

EVs differ from fossil fuel-powered vehicles in that the electricity they consume can be generated from a wide range of sources, including fossil fuels, nuclear power, and renewable sources such as solar power and wind power or any combination of that (Pistoia, 2010). The most commonly used electric vehicles are battery EVs (BEVs). BEVs are vehicles that are entirely powered by rechargeable lithium-ion batteries. Lithium-ion batteries have a higher energy and power density and longer life span, than most other practical batteries (Yoshio et al., 2009). Jaller et al. (2018) identifies BEVs as the cleanest vehicles in transportation since their rechargeable battery operates without emitting exhaust pollutants such as volatile organic compounds, hydrocarbons, carbon monoxide, ozone, lead, and various oxides of nitrogen. Additionally, BEVs can be maintained easier and with lower costs compared to the other EV types since they do not contain complex engine structures like the combustion engine, transmission, fuel tank, cooling, and exhaust system. Nevertheless, even with the improved technology, EVs have limited driving ranges, high battery costs, long recharging times, and sparse recharging facilities (Giordano et al., 2018). Throughout this dissertation the term EV will refer to a commercial BEV.

Nykvist and Nilsson (2015) reports that the battery is the most expensive part of an EV. Nowadays, the lithium-ion battery of an EV costs more than a quarter of its total price. Luckily, the prices decreased by 14% annually between 2007 and 2014. Recently, the total cost of ownership of an EV is cheaper than that of an equivalent ICEV in the EU and USA, due to lower fueling and maintenance costs. More information related to the EVs and the battery types can be found in Mangiaracina et al. (2015).

1.1. Electric Vehicle Routing Problems

The Electric Vehicle Routing Problem (EVRP) is an extension of the well-known VRP. The problem aims to determine the minimum routing cost while serving customers with known demands by utilizing a fleet of EVs. EVs are benefited in logistics operations since they have zero tailpipe emissions and their operational costs are less than ICEVs. However, they have some limitations such as limited battery capacity and charging facilities, and long charging duration. The EVRP studies targeting these challenges have recently gained momentum in the last 20 years.

Schneider et al. (2014) introduce the EVRP with Time Windows (EVRPTW) and propose a metaheuristic method to solve the problem with full recharging (EVs are fully charged when they visit a station). As well as classical EVRP and EVRPTW, many other adaptations of the problem are studied such as the electric location routing problem that make decisions on both the locations of recharging facilities and the routes of EVs (Yang and Sun, 2015; Hof et al., 2017; Schiffer and Walther, 2017a,b; Paz et al., 2018), the EVRP with a mixed fleet of EVs and ICEVs (Goeke and Schneider, 2015; Macrina et al., 2019; Hiermann et al., 2016), the EVRP with hybrid EVs (Mancini, 2017; Zhen et al., 2020), the electric dial-ride problem in which customers specify pickup and delivery requests between origins and destinations (Cordeau and Laporte, 2007; Molenbruch et al., 2017; Ho et al., 2018), the electric two-echelon VRP (Baldacci et al., 2013; Breunig et al., 2019; Jie et al., 2019), the EVRP with flexible time windows (Taş, 2021), the EVRP with fast recharging (Felipe et al., 2014; Çatay and Keskin, 2017; Keskin and Çatay, 2018), and electric pickup and delivery problem with time windows (Goeke, 2019). In addition, some EVRP extensions consider stochastic variables such as energy consumption (Pelletier et al., 2019), waiting times in stations (Keskin et al., 2019a) and availability of stations (Kullman et al., 2018; Keskin et al., 2019b, 2021). We refer the interested reader

to Qin et al. (2021) for an extensive review on the electric vehicle routing problem variants.

Additionally, the concept of flexible delivery locations attracts attention in the VRP literature in the last couple of years. Sadati et al. (2022) introduces the EVRP with Flexible Delivery (EVRP-FD) in which the customers may request their packages to be delivered to one of the locations that they specify within the related time window.

The research concerning finding the most suitable technology to handle charging needs of the commercial EVs focuses on using battery swap stations since they provide fastest service much like gas and petrol stations today (Ahmad et al., 2020; Vallera et al., 2021). EVRP with battery swap stations are also studied by many articles (e.g. Pelletier et al., 2017; Hof et al., 2017). Furthermore, another charging related research direction is related to the battery charging function. The studies mentioned by now consider recharging time to be linearly proportional to the amount of loaded energy. On the other hand, more realistic approaches like considering nonlinear battery recharging function have also been considered by several studies including Montoya et al. (2017) and Zhang et al. (2018).

The VRP is an NP-Hard combinatorial optimization problem. Therefore, the majority of the studies related to the VRP variants focus on the heuristic solution methods including constructive heuristics (Dell'Amico et al., 2007) and multi-phase heuristics (Petch and Salhi, 2003; Ganesh and Narendran, 2007) or metaheuristic approaches like Tabu Search (Xu et al., 2017; Balcik, 2017), Simulated Annealing (Vincent et al., 2017; Yassen et al., 2017) and Large Neighborhood Search (Bektaş and Laporte, 2011; Hiermann et al., 2016; Keskin and Çatay, 2018). We direct the interested reader to Cordeau et al. (2005) and Elshaer and Awad (2020) for reviews of heuristic and metaheuristic algorithms proposed for VRPs, respectively.

In this dissertation, we propose exact and heuristic methods based on the Branch-and-Price (BP) and Branch-and-Price-and-Cut (BPC) to solve the EVRPTW and the EVRP-FD. The Column Generation (CG) procedure is used to solve each node of these methodologies since it is shown to be one of the most effective methods for NP-Hard combinatorial optimization problems (Danna and Pape, 2005; Kallehauge et al., 2005; Baldacci et al., 2012). The subproblem of the CG is defined as the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). ESPPRC is solved by using label setting and label correcting algorithms, the performance of which can be improved with several different strategies including the state space augmentation algorithm (Boland et al., 2006; Righini and Salani, 2008), the *ng-route* algorithm (Baldacci et al., 2011; Andelmin and Bartolini, 2017) which allow

non-elementary paths, and the Pulse algorithm that provides label-free approach search structure (Lozano et al., 2015; Thomas et al., 2019; Cabrera et al., 2020). Valid inequalities such as Chvátal-Gomory cuts (Petersen et al., 2008), 2-path cuts (Kohl et al., 1999), and subset-row inequalities (Jepsen et al., 2008) are often used to strengthen the bounds. The performance of the CG is usually further improved with the approaches like the bidirectional search (Desaulniers et al., 2016; Thomas et al., 2019; Tilk and Goel, 2020), finding lower bounds on the reduced costs of paths (Baldacci et al., 2011; Di Puglia Pugliese and Guerriero, 2012; Lozano et al., 2015), finding upper bounds on the solution costs (Santos et al., 2011; Yu et al., 2022), and saving routes to be used in later iterations (Taş et al., 2014).

1.2. Organization

In Chapters 2 and 3, we address the EVRPTW, which is an extension of the well-known VRP with Time Windows (VRPTW). Introduced by Schneider et al. (2014) the EVRPTW deals with serving a set of customers with known demands and time windows using a homogeneous fleet of EVs dispatched from a single depot. The EVs start their routes from the depot, serve each customer exactly once, and return to the depot at the end of the day. Since they have a limited driving range, they can be recharged at charging stations en route. The stations are uncapacitated but scarce. The objective is to minimize the total distance traveled.

In Chapter 2, we propose an exact algorithm and a heuristic method to solve the EVRPTW. Both methods are based on a BPC algorithm employed with a CG procedure. CG consists of two main parts: the linear relaxation of the master problem for route selection and the pricing subproblem for route generation. Our branching strategy is to employ the first attainable rule among the following branching rules that are widely applied in the literature (Kohl et al., 1999; Kallehauge et al., 2005; Desaulniers et al., 2016): branching (i) on the total number of vehicles, (ii) on the total number of recharges, (iii) on the total number of recharges at a station, and (iv) on the total flow on an arc.

The performance of CG is enhanced by using a set of valid inequalities known as subset-row inequalities (Jepsen et al., 2008). Moreover, the BPC algorithm is improved by employing six acceleration techniques. The first method, Intermediate Column Pool (ICP), preserves routes with non-negative reduced costs which can be

used in later iterations when the dual variables change. The second technique is obtained by modifying a well-known label correcting algorithm, *ng-route* algorithm (Baldacci et al., 2011), for solving the pricing subproblem of the BPC procedure. The pricing subproblem varies between our two solution procedures as follows. In the heuristic algorithm referred to as Heuristic Column Generator (HCG), the labeling algorithm is obtained by relaxing the *ng-route* algorithm. On the other hand, the pricing subproblem of the exact algorithm is solved with a sequential procedure including heuristic and exact labeling methods. The third procedure that we implement is the bidirectional labeling algorithm, which is shown to be quite effective for solving shortest path problems (Pohl, 1971; Luby and Ragde, 1989; Righini and Salani, 2006). The fourth acceleration method is denoted as Augmented Node (AN) and benefits from the domination of stations that can be visited between each pair of customers (Bruglieri et al., 2016). More specifically, this method reduces the problem network by eliminating the dominated stations dynamically and explores the search tree by traveling from one customer to another either directly or via a station. The fifth method provides lower bounds on reduced costs of the paths on each node, which allows the removal of suboptimal routes before they arrive at the depot (Baldacci et al., 2011). Lastly, we present the Integer Master Problem (IMP) that provides good upper bounds for the problem at the root node of the BPC tree.

In the computational analysis, we evaluate the proposed procedures by using the benchmark instances provided by Desaulniers et al. (2016) and present solutions for several instances that have not been solved before.[1]

In Chapter 3, we solve the EVRPTW by utilizing the Pulse algorithm for the pricing subproblem of the BP procedure. The Pulse algorithm simplifies the difficulties of the classical labeling algorithms such as label storage and dominance (Lozano et al., 2015). Its depth-first search structure avoids the storage of an excessive number of labels at the same time. The Pulse algorithm consists of several strategies which fathom suboptimal paths. One of these strategies is the bounding method which creates a bound matrix using lower bounds on the reduced costs found for each node and discrete values of resource consumption. The bound matrix is beneficial to eliminate unfavorable paths and thereby speeding up the process. In addition, another method called rollback pruning simplifies the dominance procedure, often used in labeling algorithms (Feillet et al., 2004; Kohl et al., 1999), by evaluating whether the last node included in the path should be removed. The construction of this strategy allows the disposal of most of the dominance rules.

The BP procedure with the Pulse algorithm is improved using several acceleration

[1]The related solutions have been reported in Duman et al. (2021).

techniques including the IMP, the ICP, and the AN methods. Additionally, each iteration of the Pulse algorithm is prematurely terminated when the number of complete routes with negative reduced cost reaches a threshold value. We provide three rules including branching (i) on the total number of vehicles, (ii) on the total number of recharges, and (iii) on the total flow value of an arc.

We provide the results of the computational experiments evaluating the performance of the BP algorithm on Desaulniers et al. (2016) data set again.. We compare the results with those obtained in Chapter 2 and present several new solutions.

In Chapter 4, we address the Electric Vehicle Routing Problem with Flexible Deliveries (EVRP-FD) introduced by Sadati et al. (2022) and propose an effective BP algorithm to solve it. In the EVRP-FD, customers may request their orders to be delivered to one of the predetermined delivery locations within the corresponding time window. Each day a homogeneous fleet of EVs is dispatched from a central depot to serve each customer exactly once in one of their locations within the related time window. The problem aims to minimize the total distance traveled. We provide a Mixed-Integer Linear Programming (MILP) formulation for the EVRP-FD and develop a BP algorithm employed with a CG procedure. A bidirectional Pulse algorithm is proposed to solve the pricing subproblem of the EVRP-FD. The BP algorithm is promoted by employing several acceleration techniques: the bidirectional search, IMP, ICP, and heuristic pricing. We start each iteration of the CG after employing the HCG. The performance of the proposed algorithm is tested using data adapted from the literature.

In Chapter 5, we conclude the dissertation with final remarks and provide directions for future research in this area.

2. BRANCH-AND-PRICE-AND-CUT METHODS FOR ELECTRIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

In this chapter, we address the Electric VRP with Time windows (EVRPTW), which is an extension of the well-known VRP with Time Windows (VRPTW) and introduced by Schneider et al. (2014). The EVRPTW deals with serving a set of customers with known demands and time windows using a homogeneous fleet of electric vehicles (EVs) dispatched from a single depot. The EVs start their routes from the depot, serve each customer exactly once, and return to the depot at the end of the day. Since they have limited driving range, they can be recharged at charging stations en-route. The stations are uncapacitated but scarce. The objective is to minimize the total distance traveled. Schneider et al. (2014) assumed a full-recharge policy, which is later relaxed in the follow-up studies (e.g, Bruglieri et al., 2015; Desaulniers et al., 2016; Keskin and Çatay, 2016; Hiermann et al., 2016).

To the best of our knowledge, Desaulniers et al. (2016) is the only study that proposed exact methods for solving the EVRPTW. In this chapter, we develop an exact algorithm and a heuristic method for the same problem. Both methods are based on a Branch-and-Price-and-Cut (BPC) algorithm employed with a Column Generation (CG) procedure. CG consists of two main parts: the linear relaxation of the master problem for route selection and the pricing subproblem for route generation. The performance of CG is enhanced by using a set of valid inequalities known as subset-row inequalities (Jepsen et al., 2008). Moreover, the BPC algorithm is improved by employing six acceleration techniques. The first method, effectively employed by Taş et al. (2014), preserves routes with non-negative reduced costs which can be used in later iterations when the dual variables change. The second technique is obtained by modifying a well-known label correcting algorithm, *ng-route* algorithm (Baldacci et al., 2011), for solving the pricing subproblem of the BPC procedure. The pricing subproblem varies between our two solution procedures as follows. In the heuristic algorithm referred to as Heuristic Column Generator (HCG), the labeling algorithm is obtained by relaxing the *ng-route* algorithm. On the other hand,

the pricing subproblem of the exact algorithm is solved with a sequential procedure including heuristic and exact labeling methods. The third procedure that we implement is bidirectional labeling algorithm, which is shown to be quite effective for solving shortest path problems (Pohl, 1971; Luby and Ragde, 1989; Righini and Salani, 2006). The fourth acceleration method is based on a branching strategy only on customer nodes and benefits from the domination of stations that can be visited between each pair of customers (Bruglieri et al., 2016). More specifically, this method reduces the problem network by eliminating the dominated stations dynamically and explores the search tree by traveling from one customer to another either directly or via a station. The fifth method provides lower bounds on reduced costs of the paths on each node, which allows the removal of suboptimal routes before they arrive at the depot (Baldacci et al., 2011). Lastly, we present a method providing good upper bounds for the problem at the root node of the BPC tree. The contributions of this chapter to the existing literature can be summarized as follows.

- We adopt the state-of-the-art methods from the literature to develop an effective BPC algorithm. In addition, we introduce an acceleration method which benefits from network reduction through the elimination of dominated stations. This method enables us to consider only customers during route generation by creating an *augmented node* which combines a customer and a station node.
- We first develop an iterative heuristic algorithm based on a relaxation in the pricing subproblem. This method is then used to obtain an effective BPC by generating a CG procedure starting with the heuristic algorithm and performing an exact labeling algorithm next.
- We provide computational analysis to evaluate the proposed procedures using instances with up to 100 customers and 21 stations, and present solutions for several instances that have not been solved before.

The remainder of this chapter is organized as follows. Section 2.1 reviews the related literature. Section 2.2 describes the problem and formulates its mathematical programming model. Section 2.3 presents the BPC procedures. Section 2.4 details the experimental study and discusses the numerical results. Finally, concluding remarks are provided in Section 2.5.

Artmeier et al. (2010) is the first study that considers an EV within the context of planning a single route. Then, the Recharging VRP is introduced by Conrad and Figliozzi (2011), where the fleet consists of EVs that can be recharged at a subset of customer locations during their service. Erdoğan and Miller-Hooks (2012) investigate the Green VRP (GVRP) by considering a homogeneous fleet of AFVs. The authors assume that the vehicles are fully refuelled at constant time and employ two heuristic approaches to solve the problem.

The EVRPTW is introduced by Schneider et al. (2014). The authors assume that the EVs are fully recharged at the stations and the recharging time is proportional to the amount of energy transferred. They develop three algorithms by integrating Variable Neighborhood Search (VNS) with Tabu Search (TS) and compare their performances using small- and large-size data sets that they generate based on the VRPTW instances of Solomon (1987). Inspired by this study, Keskin and Çatay (2016) relax the full charge restriction by allowing partial recharges at stations and propose an Adaptive Large Neighborhood Search (ALNS) algorithm to solve it. The problem with partial recharging is also addressed by Bruglieri et al. (2015). The authors formulate the Mixed-Integer Linear Programming (MILP) model of the problem, develop a VNS algorithm, and evaluate their performances on small-size instances.

Various variants of the Electric VRP (EVRP) have been studied in the literature including the mixed fleet which consists of both EVs and Internal Combustion Engine Vehicles (ICEVs) (Goeke and Schneider, 2015; Macrina et al., 2019; Hiermann et al., 2016), hybrid EVs (Zhen et al., 2020); location routing problem with EVs (Yang and Sun, 2015; Hof et al., 2017; Schiffer and Walther, 2017a,b; Paz et al., 2018), nonlinear battery recharging function (Montoya et al., 2017; Zhang et al., 2018), flexible time windows (Taş, 2021), fast recharging (Felipe et al., 2014; Çatay and Keskin, 2017; Keskin and Çatay, 2018) and battery swap stations (Pelletier et al., 2017; Hof et al., 2017). In addition, some EVRP extensions consider stochastic variables such as energy consumption (Pelletier et al., 2019), waiting times in stations (Keskin et al., 2019a) and availability of stations (Kullman et al., 2018; Keskin et al., 2019b, 2021).

There are only few studies in the literature that solve EVRPs with exact methods. However, several VRP extensions are solved by exact solution methods based on the branch-and-bound. The interested reader is referred to Baldacci et al. (2012) and Semet et al. (2014) for the solution methods related to VRP, to Coelho and

Laporte (2013) and Grønhaug et al. (2010) for the algorithms proposed to solve inventory routing problems, and to Yuan et al. (2015) for the approaches developed to solve the home health care routing problem. Among the studies that propose efficient procedures, Taş et al. (2014) develop a Branch-and-Price (BP) method for the VRP with stochastic travel times and soft time windows. The subproblem of the CG is defined as the Elementary Shortest Path Problem with Resource Constraints (ESPPRC) and the authors benefit from the state space augmentation algorithm (Boland et al., 2006; Righini and Salani, 2008) to solve the ESPPRC. Andelmin and Bartolini (2017) consider a BP method for the GVRP. The authors describe several different valid inequalities and best results are achieved with Chvátal-Gomory cuts. Furthermore, they provide modifications in the CG method to increase the efficiency of the subproblem such as implementing the *ng-route* algorithm. The *ng-route* algorithm improves the performance of the BPC procedure by allowing non-elementary paths. For a partial path at node i, the algorithm defines a set of customers which are forbidden to be visited more than once by that path. These dynamic sets are obtained from the *ng-sets* that are defined for each node and contains neighbors of that node, often the ones in short traveling distances. This method is also applied by Desaulniers et al. (2015) and Rothenbächer et al. (2016) to solve particular VRP variants.

Desaulniers et al. (2016) address four different versions of the problem in two categories, and solve them with a branch-and-price-and-cut method. The authors implement a set partitioning formulation for the relaxed master problem and solve the pricing subproblem using a labeling correcting algorithm. Moreover, acceleration strategies such as bidirectional search, the *ng-route* relaxation, and heuristic relaxation in the network are applied. In this chapter, we develop two BPC algorithms with an exact algorithm and a heuristic column generation method to solve the EVRPTW by considering multiple and single partial recharges. We allow partial recharges since it constitutes a realistic setting, and is more challenging to solve.

2.2. Problem Description and Model

The EVRPTW is defined on the digraph $G = (N, A)$, where $N = \{0, 1, ..., n + 1\}$ is the set of nodes and A is the set of arcs. The notation provided by Keskin and Çatay (2016) is benefited for the problem formulation. In set N, node 0 and node $n + 1$ represent the departure depot and the arrival depot, respectively. The set of

customers is denoted by $V = \{1, ..., n\}$. The demand of each customer i given by q_i should be satisfied within the specified time window $[e_i, l_i]$, and the service duration is s_i for all $i \in V$. If an EV arrives at customer i before the earliest possible service time e_i, it waits until e_i. On the other hand, arrivals after the latest possible service time l_i are not permitted. The set of nodes including customers and the departure depot 0 is denoted by $V_0 = V \cup \{0\}$, customers and the arrival depot is represented by $V_{n+1} = V \cup \{n+1\}$, and customers, the arrival and departure depots is provided by $V_{0,n+1} = V_0 \cup \{n+1\}$. The fleet consisting of identical EVs with cargo capacity C and battery energy capacity Q is represented by K. Note that there is no limit on the number of EVs in the fleet. The battery charging rate of each EV is g and the consumption rate is h.

Table 2.1. Mathematical notation

Sets	
V	Set of customers, $V = \{1, .., n\}$
V_0	Set of customers and the departure depot, $V_0 = V \cup \{0\}$
V_{n+1}	Set of customers and the arrival depot, $V_{n+1} = V \cup \{n+1\}$
$V_{0,n+1}$	Set of customers, the departure depot and the arrival depot, $V_{0,n+1} = V_0 \cup \{n+1\}$
F	Set of recharging stations
N	Set of all nodes, $N = V_{0,n+1} \cup F$
K	Set of EVs
Parameters	
d_{ij}	Distance along arc $(i, j) \in A$
t_{ij}	Travel time along arc $(i, j) \in A$
q_i	Demand of customer i, $i \in V$
s_i	Service time spent at customer i, $i \in V$
$[e_i, l_i]$	Service time window at node i, $i \in V_{0,n+1}$
C	Vehicle freight capacity
Q	Vehicle battery capacity
g	Battery charging rate
h	Energy consumption rate
t_{ijs}	Detour time spent for visiting station s while traveling from node i to node j, $t_{ijs} = t_{is} + t_{sj} - t_{ij}$, $i \in V_0$, $j \in V_{n+1}$, $s \in F$
d_{ijs}	Detour distance traversed by visiting station s while traveling from node i to node j, $d_{ijs} = d_{is} + d_{sj} - d_{ij}$, $i \in V_0$, $j \in V_{n+1}$, $s \in F$
Decision Variables	
τ_i^k	Service start time of vehicle k at customer i, $i \in V$, $k \in K$
y_i^k	Battery state of charge (SoC) of vehicle k at its arrival at node i, $i \in V_{0,n+1}$, $k \in K$
Y_{ijs}^k	Battery SoC of vehicle k at its departure from station s if it travels from node i to node j through station s, $i \in V_0$, $j \in V_{n+1}$, $s \in F$, $k \in K$
y_{ijs}^k	Battery SoC of vehicle k at its arrival at station s if it travels from node i to node j through station s, $i \in V_0$, $j \in V_{n+1}$, $s \in F$, $k \in K$
x_{ij}^k	1, if vehicle k traverses arc (i, j), directly or through a station, 0 otherwise, $i \in V_0$, $j \in V_{n+1}$, $k \in K$
z_{ijs}^k	1, if vehicle k travels from node i to node j through station s, 0 otherwise, $i \in V_0$, $j \in V_{n+1}$, $s \in F$, $k \in K$

The distance along arc (i, j) is denoted by d_{ij} where the triangular inequality is satisfied for each arc $(i, j) \in A$. However, in case a vehicle visits a station s while traveling from node i to node j we introduce d_{ijs} that represents the total distance traversed as $d_{ijs} = d_{is}+d_{sj}-d_{ij}$. Additionally, t_{ij} denotes the time spent for traveling directly from node i to node j. Similarly, $t_{ijs} = t_{is}+t_{sj}-t_{ij}$ represents the traveling time when station s is visited between node i and node j.

The decision variables τ_i^k and y_i^k keep track of the start time of service and the battery state of charge (SoC) of vehicle k at its arrival at node i, respectively. In case vehicle k travels from node i to node j through station s, the decision variables y_{ijs}^k and Y_{ijs}^k represent the battery SoC of that vehicle at its arrival at station s and departure from station s, respectively. The binary decision variable x_{ij}^k takes the value 1 if vehicle k traverses arc (i, j) directly or through a station, and 0 otherwise. Furthermore, z_{ijs}^k is a decision variable which takes the value 1 if vehicle k traverses arc (i, j) through station s, and 0 otherwise. The mathematical notation is summarized in Table 2.1. .

EVs are dispatched from the depot at the beginning of each day with their batteries fully charged overnight, and they return to the depot after they complete their tour. The fleet must respect the customer time windows and the scheduling horizon represented by the time window of the depot. The stations are identical in terms of charging speed and cost, and can be visited multiple times by the same or different EVs. For the sake of simplicity, we assume that the battery charging time is linearly proportional to the amount of energy transferred. In addition, we allow at most one recharge between each pair of customers. This is a rational assumption since recharging the battery of EVs at multiple stations when it travels from one customer to another may not be practical in urban logistics operations.

Bruglieri et al. (2016) developed a new formulation for the GVRP which models the visits to stations implicitly. Below, we present a modified version of this formulation for solving the EVRPTW which benefits from arc-based decision variables.

$$\min \quad \sum_{i \in V_0} \sum_{j \in V_{n+1}, j \neq i} \sum_{k \in K} \left(d_{ij} x_{ij}^k + \sum_{s \in F} d_{ijs} z_{ijs}^k \right) \tag{2.1}$$

subject to

$$\sum_{k\in K}\sum_{j\in V_{n+1},j\neq i} x_{ij}^k = 1, \qquad i \in V, \quad (2.2)$$

$$\sum_{i\in V_0,i\neq j} x_{ij}^k - \sum_{i\in V_{n+1},i\neq j} x_{ji}^k = 0, \qquad k \in K, j \in V_{0,n+1}, j \neq i, \quad (2.3)$$

$$\sum_{s\in F} z_{ijs}^k \leq x_{ij}^k, \qquad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.4)$$

$$\tau_i^k + (t_{ij} + s_j)x_{ij}^k + \sum_{s\in F}\Big(t_{ijs}z_{ijs}^k + g(Y_{ijs}^k - y_{ijs}^k)\Big) - l_0(1 - x_{ij}^k) \leq \tau_j^k, \quad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.5)$$

$$e_j \leq \tau_j^k \leq l_j, \qquad k \in K, j \in V_{0,n+1}, \quad (2.6)$$

$$\sum_{i\in V} q_i \sum_{j\in V_{n+1},j\neq i} x_{ij}^k \leq C, \qquad k \in K, \quad (2.7)$$

$$0 \leq y_j^k \leq y_i^k - hd_{ij} + (Q + hd_{ij})\Big(1 - x_{ij}^k + \sum_{s\in F} z_{ijs}^k\Big), \quad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.8)$$

$$y_j^k \leq \sum_{s\in F}(Y_{ijs}^k - hd_{sj}z_{ijs}^k) + Q\Big(1 - \sum_{s\in F} z_{ijs}^k\Big) + Q(1 - x_{ij}^k), \quad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.9)$$

$$y_{ijs}^k \leq y_i^k - hd_{is}z_{ijs}^k + Q(1 - x_{ij}^k), \qquad k \in K, s \in F, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.10)$$

$$0 \leq y_{ijs}^k \leq Y_{ijs}^k \leq Qz_{ijs}^k, \qquad k \in K, s \in F, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.11)$$

$$\sum_{s\in F}\sum_{i\in V_0}\sum_{j\in V_{n+1},j\neq i} z_{ijs}^k \leq 1 \qquad k \in K, \quad (2.12)$$

$$x_{ij}^k \in \{0, 1\}, \qquad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (2.13)$$

$$z_{ijs}^k \in \{0, 1\}, \qquad k \in K, i \in V_0, j \in V_{n+1}, s \in F, i \neq j. \quad (2.14)$$

The objective (2.1) is to minimize the total distance traveled. Constraints (2.2) guarantee that each customer is visited exactly once. Flow balance is attained by constraints (2.3). Constraints (2.4) ensure that at most one station is visited between each pair of customers. Constraints (2.5) enforce the time feasibility of arcs emanating from the customers and the depot. Constraints (2.6) make sure that each node is served within its time window. Additionally, constraints (2.7) ensure that the vehicle capacity is not exceeded. Battery SoC consistency throughout the route is satisfied by constraints (2.8) - (2.10). In particular, constraints (2.8) guarantee the battery feasibility at customers and at the depot. If arc $(i, j) \in A$ is not traveled or if a station is visited between nodes i and j, these constraints are redundant. Constraints (2.9) guarantee the battery feasibility at node j if a station is visited between node i and node j, and determine the amount of energy transferred to the battery. Battery feasibility at the arrival to the stations is provided by constraints

(2.10). Moreover, if a station is not visited constraints (2.11) guarantee that the related charge variables take the value of 0. Constraints (2.12) are included in the single-recharge case to limit the number of recharges on each route by one. Constraints (2.13) and (2.14) define the binary decision variables.

2.3. Branch-and-Price-and-Cut

Before the BPC procedure starts, we employ a preprocessing method which removes infeasible arcs from the network. The method suggested by Schneider et al. (2014) removes the arc (i, j) if the vehicle battery capacity is not enough to visit node j after node i, or if the vehicle arrives at node j after visiting node i later than the time window of node j closes, or if the total time of serving nodes i and j and then traveling to the arrival depot exceeds the closing time of the window at the depot, or if it is not possible to visit any station or depot before and after traversing that arc with full battery capacity.

We obtain an initial solution using the construction algorithm suggested by Keskin and Çatay (2016). In this greedy algorithm, the routes are obtained by adding the customer with the minimum insertion cost to the route, until no more insertions can be performed without violating the load and the time window constraints. Battery feasibility is satisfied by embedding the station with the minimum insertion cost when needed.

In the remaining of this section, we present detailed descriptions of the CG procedure, the branching strategy, the implementation of a set of well-known valid inequalities, acceleration methods employed to improve the performance of the algorithm, and the heuristic algorithm.

2.3.1. Column Generation

The master problem of the CG is represented by the set partitioning formulation (2.15)-(2.17). In this formulation, the set of all feasible routes starting from the departure depot (node 0) and ending at the arrival depot (node $n+1$) is represented by Ω. Parameter c_p denotes the cost of route p, a_{pi} is equal to 1 if route p visits

node i, and 0 otherwise. Moreover, θ_p is a binary decision variable which takes the value of 1 if route p is part of the solution, and 0 otherwise.

$$\min \quad \sum_{p \in \Omega} c_p \theta_p \tag{2.15}$$

$$\text{subject to} \quad \sum_{p \in \Omega} a_{pi} \theta_p = 1, \qquad i \in V, \tag{2.16}$$

$$\theta_p \in \{0, 1\}, \qquad p \in \Omega. \tag{2.17}$$

The objective function (2.15) minimizes the total cost. Constraints (2.16) make sure that each customer is visited exactly once, and constraints (2.17) define the binary decision variables. Only a subset of routes Ω_R is evaluated at each iteration in the Linear Programming (LP) relaxation of the set partitioning formulation, called Restricted Linear Programming Master Problem (RLPMP). The interested reader is referred to Kallehauge et al. (2005) for the details about CG methods proposed for the VRP.

$$\min \quad \bar{c}_p \tag{2.18}$$

$$\text{subject to} \quad (2.2) - (2.14) \tag{2.19}$$

The pricing subproblem for each vehicle k, defined by the MILP formulation (2.18)-(2.19), corresponds to an ESPPRC. The elementary property is emphasized on customer nodes specifically since each customer has to be visited exactly once. The CG starts by solving the RLPMP where only the routes of an initial feasible solution are included in Ω_R. The subproblem aims at producing routes with negative reduced cost that can improve the current solution and is solved next by using the values of the dual variables obtained by solving the RLPMP. The new routes are added into the RLPMP that is then reoptimized. This procedure is repeated until the subproblem cannot find any route with negative reduced cost. The reduced cost $\bar{c}_p$ associated with route p is calculated as $\bar{c}_p = c_p - \sum_{i \in V} a_{pi} \pi_i$, where π_i is the value of the dual variables associated with constraints (2.16).

2.3.1.1. A Generic labeling Algorithm for the ESPPRC

We implement a modified version of the label correcting algorithm developed by Feillet et al. (2004) and Desaulniers et al. (2016). In the forward labeling algorithm, a label is associated with each partial path p from the departure depot to node i, $i \in N$. A label consists of several components representing the consumption of resources. In our problem partial recharging is allowed, thus when a station is visited the recharge quantity should be determined and the recharging time should be added into the route time. We adopt three label components to define the total route time under different conditions, T_i^{tMin}, T_i^{tMax}, and T_i^{rtMax} (Desaulniers et al., 2016). T_i^{tMin} denotes the earliest possible arrival time to node i, and if a station is visited prior to node i by partial path p, the minimum amount of energy ensuring the battery feasibility up to node i is considered. If no station is visited, the value of T_i^{tMin} is equal to the summation of the travel times of each arc traversed along path p and the service times of each customer visited by path p. T_i^{tMax} is the earliest possible arrival time to node i, and if a station is visited prior to node i, the maximum recharging duration guaranteeing time window feasibility is considered. If no station is visited, T_i^{tMax} is similarly computed by including the travel times of the arcs traversed and the service times spent at customer locations. Despite the fact that EVs can never be recharged at customer locations, we make the artificial assumption that EVs can be recharged at customer locations to define T_i^{rtMax}. Using this assumption, T_i^{rtMax} is the maximum time required to fully recharge the battery at node i. If a station is visited along path p before node i, T_i^{rtMax} considers that the minimum amount of energy guaranteeing the battery feasibility is loaded. The total freight load of the path up to node i is denoted by T_i^{load}. a_p denotes the number of customers that are unreachable by path p where customer j is said to be unreachable if $T_i^{load} + q_j > C$ or $T_i^{tMin} + t_{ij} + s_i > l_j$. In addition, $\mathbf{V}_p^V$ is a vector of binary variables where V_p^j states that node j is unreachable by path p if its value is equal to 1. The interested reader is referred to Feillet et al. (2004) for more details about unreachable nodes. Altogether, a state of path p is defined with its reduced cost $\bar{c}_p$ and label $L_p = (T_i^{load}, T_i^{tMin}, T_i^{tMax}, T_i^{rtMax}, a_p, \mathbf{V}_p^V)$.

The time related label components T_p^{tMin}, T_p^{tMax} and T_p^{rtMax} are sufficient to describe minimum and maximum feasible recharges at stations. Thus, at each node time window feasibility and battery SoC feasibility can be maintained through these labels.

The extension of a label from node i to node j is performed by using the following label calculating equations (2.20)-(2.33) where $|S_p|$ is the number of stations visited

along path p and S_{ij} is defined as the slack time between the earliest possible service start time e_j and the earliest possible arrival time to node j. If slack time exists, it refers to the availability of (additional) recharging time at stations. In addition, R_{ij} denotes the minimum required recharging time to maintain battery feasibility minus the slack time and calculated as $R_{ij} = \max\Big(0, \max\Big(0, T_i^{rtMax} - S_{ij}\Big) + gd_{ij} - gQ\Big)$.

$$T_j^{load} = T_i^{load} + q_j \tag{2.20}$$

$$T_j^{tMin} = \begin{cases} \max\Big(e_j, T_i^{tMin} + t_{ij} + s_i\Big), & \text{if } |S_p| = 0 \\ \max\Big(e_j, T_i^{tMin} + t_{ij} + s_i\Big) + R_{ij}, & \text{otherwise} \end{cases} \tag{2.21}$$

$$T_j^{tMax} = \begin{cases} \min\Big(l_j, \max(e_j, T_i^{tMin} + T_i^{rtMax} + t_{ij} + s_i)\Big), & \text{if } i \in F \\ \min\Big(l_j, \max(e_j, T_i^{tMax} + t_{ij} + s_i)\Big) & \text{otherwise} \end{cases} \tag{2.22}$$

$$T_j^{rtMax} = \begin{cases} T_i^{rtMax} + gd_{ij}, & \text{if } |S_p| = 0 \\ \min\Big(gQ, \max(0, T_i^{rtMax} + S_{ij} + gd_{ij})\Big) & \text{otherwise} \end{cases} \tag{2.23}$$

where

$$S_{ij} = \begin{cases} \max\Big(0, \min(e_j - (T_i^{tMin} + t_{ij} + s_i\Big), T_i^{rtMax})\Big) & \text{if } i \in F \\ \max\Big(0, \min(e_j - (T_i^{tMin} + t_{ij} + s_i), T_i^{tMax} - T_i^{tMin})\Big) & \text{otherwise} \\ \end{cases}$$

At any step of the algorithm, the feasibility of the routes with respect to the load and time constraints is maintained. If $T_j^{load} > C$, $T_j^{tMin} > l_j$, $T_j^{tMin} > T_j^{tMax}$, or $T_j^{rtMax} > Qg$, the partial path p is infeasible and thus eliminated. The EVRPTW with single recharging requires an additional feasibility rule, $|S_p| \leq 1$, that limits the number of station visits by one.

Next, we explain the rules used to eliminate dominated paths. Note that the superscript i is removed from the label components of paths p_1 and p_2 since both paths arrive at the same node. definition definition

Definition 1 *(Dominance Relation) Let p_1 and p_2 be two distinct partial paths from the departure depot to node i and their labels be $(L_{p_1}, \bar{c}_{p_1})$ and $(L_{p_2}, \bar{c}_{p_2})$, respectively. Path p_1 dominates path p_2 if and only if:*

$$\bar{c}_{p_1} \leq \bar{c}_{p_2} \tag{2.24}$$
$$W_{p_1}^{load} \leq W_{p_2}^{load}, \tag{2.25}$$
$$T_{p_1}^{tMin} \leq T_{p_2}^{tMin}, \tag{2.26}$$
$$a_{p_1} \leq a_{p_2}, \tag{2.27}$$
$$V_{p_1}^{j} \leq V_{p_2}^{j} \text{ for all } j \in V \tag{2.28}$$
$$T_{p_1}^{rtMax} - (T_{p_1}^{tMax} - T_{p_1}^{tMin}) \leq T_{p_2}^{rtMax} - (T_{p_2}^{tMax} - T_{p_2}^{tMin}) \tag{2.29}$$
$$T_{p_1}^{rtMax} + T_{p_1}^{tMin} \leq T_{p_2}^{rtMax} + T_{p_2}^{tMin} \tag{2.30}$$

and at least one of the above inequalities is strictly satisfied.

Inequality (2.24) indicates that the reduced cost of path p_1 is less than or equal to the reduced cost of path p_2. Inequality (2.25) provides that the load capacity of the vehicle is used less by path p_1 compared to path p_2. Similarly, inequality (2.26) states that path p_1 arrives earlier compared to path p_2. Inequality (2.27) expresses that the number of unreachable nodes by path p_1 is less than or equal to that by path p_2. In addition, inequality (2.28) implies that every unreachable node for path p_1 is also unreachable for path p_2.

If path p_1 does not visit any station, $T_{p_1}^{tMax}$ is equal to $T_{p_1}^{tMin}$, and thus the difference $(T_{p_1}^{tMax} - T_{p_1}^{tMin})$ is equal to zero. On the other hand, if at least one station is visited, this difference denotes the additional available recharging time that can be added to the minimum required recharging time. Since $T_{p_1}^{rtMax}$ states the current battery SoC in terms of the required time for recharging, the difference $T_{p_1}^{rtMax} - (T_{p_1}^{tMax} - T_{p_1}^{tMin})$ measures the additional time required to fully recharge the battery, after all available time is used for recharging. Therefore, the condition (2.29) holds if path p_1 requires less time to fully recharge compared to path p_2 or alternatively the remaining energy level of path p_1 is more than that of path p_2.

Finally, inequality (2.30) states that path p_1 spends less time compared to path p_2 for recharging and traveling. This condition becomes redundant if no station is visited along the path, as it can be obtained by summing up inequalities (2.26) and (2.29).

2.3.2. Branching

If the optimal solution obtained by the CG algorithm to the LP relaxation of the formulation (2.15)-(2.17) is fractional, this solution corresponds to a lower bound

(LB) for the original problem and integer solutions are generated through a BPC tree. We employ four main branching strategies widely applied in the literature (Kohl et al., 1999; Kallehauge et al., 2005; Desaulniers et al., 2016): branching (i) on the total number of vehicles, (ii) on the total number of recharges, (iii) on the total number of recharges at a station, and (iv) on the total flow on an arc. When the CG ends with a fractional solution, using the provided order of branching rules we employ the first attainable rule.

If the first three strategies are not available we branch on flow variables as follows: Let Ω_R be the set all routes generated by the CG algorithm and $p \in \Omega_R$. The total flow for each arc is then calculated as $f_{ij} = \sum_{k \in K} x_{ij}^k$ where the values of x_{ij}^k are obtained by using the solution of the restricted linear programming master problem. More specifically, f_{ij} can also be defined as $f_{ij} = \sum_{p \in \Omega_R:(i,j) \in p} \theta_p$, where θ_p takes the value 1 if route p is in the solution and 0 otherwise. If there are more than one arc incurring fractional flow value, the arc with a flow value closest to 0.5 is selected for branching and two branches are generated as follows: f_{ij} is forced to 0 by eliminating the related arc from the graph and to 1 by deleting arcs (i, v) and (v, j) for any $v \in N \backslash \{i, j\}$.

The depth-first search strategy is implemented on the tree since it is shown to be more effective (Taş et al., 2014; Desaulniers et al., 2016). At each tree node, the CG algorithm is solved. If the solution is integral, then branching is completed on that node. A tree node producing an infeasible solution is fathomed.

2.3.3. Cuts

Valid inequalities are often used to obtain stronger bounds to the branch-and-bound tree and to improve its linear relaxations (see Desaulniers et al., 2008, 2016). So, we implement a separation algorithm using the subset-row inequalities (Jepsen et al., 2008). The subset-row inequalities are Chvátal-Gomory inequalities of rank 1 defined over subsets of the set partitioning constraints of the RLPMP. We consider only the subsets including three customers as in Jepsen et al. (2008) and Desaulniers et al. (2016). Given a subset of three customers $U \subset N$, a subset-row inequality ensures that any feasible integer solution can contain at most one route visiting two or three customers in U. The corresponding subset-row inequality is $\sum_{p \in \Omega} m_p^U \theta_p \leq 1$, where $m_p^U = \lfloor \beta_p^U / 2 \rfloor$ and β_p^U is equal to the number of visits to any customer in U along route p. The decision variable θ_p takes the value 0 if path p is not part of the solution and 1 otherwise as in formulation (2.15)-(2.17). If p is an elementary

path, β_p^U can at most be three. The separation of the subset-row inequalities starts with enumerating all subsets of three customers. Next, the algorithm checks for each subset whether the corresponding inequality is violated. Whenever violated inequalities are found, they are added to the RLPMP. One difficulty of these cuts is that the dual variables associated with the subset-row cuts cannot be integrated directly to the reduced arc costs. Instead, the dual variable associated with a subset-row cut is sent to the labeling algorithm that solves the pricing subproblem. The dual variable is then subtracted from the reduced cost of a partial path each time it visits two customers defining the cut. An additional label component needs to be included into the labeling algorithm to count the number of visits to each subset of customers.

The subset-row inequalities are proven to improve the bounds. They substantially reduce both the number of nodes to explore in the search tree and the computation time. On the other hand, the cuts may increase the problem complexity due to the integration of dual variables. We address this issue by bounding the number of cuts that can be added simultaneously to the RLPMP using two parameters: a cut can be chosen if the violation is greater than threshold value t^{SRC} and a maximum of n_{max}^{SRC} cuts can be added in each round.

2.3.4. Acceleration Techniques

Our preliminary tests revealed that the RLPMP is solved within few seconds even for larger instances. However, our exact algorithm slows down significantly during route generation. In this section, we provide methods improving the performance of the algorithm mainly by reducing the computational effort spent for solving the pricing subproblem.

The labeling algorithm is terminated prematurely whenever the number of efficient elementary routes with negative reduced costs is greater than or equal to a predetermined threshold value, n^{rt}. This approach is particularly effective when the number of feasible routes is quite large.

In what follows, we describe the acceleration techniques that we implement to enhance the performance of BPC algorithm.

2.3.4.1. Intermediate Column Pool

An Intermediate Column Pool (ICP) is implemented to store complete routes that have not been sent to the RLPMP (Taş et al., 2014). At each iteration of the CG, reduced costs of the columns in the ICP are computed with respect to the optimal dual values. If the reduced cost of any column is negative, the corresponding route is added to the RLPMP. The implementation of the column pool involves two parameters: the first one bounds the number of routes stored in the pool, n^{col}, whereas the second one limits the maximum number of iterations that each column is stored in the pool, n^{iter}.

2.3.4.2. Two-Phase Algorithm with *Ng-Route*

The *ng-route* algorithm is introduced by Baldacci et al. (2011) for solving the VRPTW. The algorithm relaxes elementary constraint of the pricing subproblem and allows selected sets of customers, *ng-sets*, to be revisited. The *ng-set* for node j, NG_j, denotes the set of ν geographically closest customers to node j where ν is the predetermined size of the neighborhood. At each step of the subproblem, the nodes belonging to the *ng-set* are allowed to be visited at most once, while others can be visited multiple times. For instance, if $i \notin NG_j$, then any route generated by the algorithm is allowed to include the cycle $i - ... - j - i$. On the other hand, a route containing such a cycle cannot be part of the optimal solution since the distance d_{ij}, already greater than the distance between node j and any other customer in NG_j, is added to the path cost although customer i has already been visited.

We modify the *ng-route* algorithm for solving the EVRPTW as outlined in Algorithm 2.1. All labels provided in Section 2.3.1.1 remain the same except the one storing unreachable nodes. Since some customers may be visited multiple times, the definition of $\mathbf{V}_p^V$ is extended to include integer values greater than one. In the algorithm, the list of nodes to be treated is denoted by I, the list of labels on node i is Π_i, and H_{ij} corresponds to the set of labels extended from node i to node j. `Extend()` procedure used in Algorithm 2.1 is detailed in Algorithm 2.2. `Dominance()` procedure represents the elimination of dominated routes using the relations provided by inequalities (2.24)-(2.30). Note that the dominance rules provided in Section 2.3.1.1 remain the same in the *ng-route* algorithm.As defined earlier, Ω_R denotes the restricted set of the routes with negative reduced costs. Note also that we eliminate non-elementary routes at the end of every iteration

and send only the elementary routes to the RLPMP. If the algorithm yields only non-elementary routes with negative reduced costs at the arrival depot, the customer with the highest multiplicity is added to *ng-sets* of each customer and the CG algorithm continues its operations using the updated *ng-sets*. Consequently, the optimality is guaranteed and a method for extracting cycles from non-elementary routes is not needed.

Algorithm 2.1: *Ng-route* algorithm

```
1  Initialize;
2  L_0 = (0, 0, 0, 0, 0, 0);
3  for I = {1, ..., N} do
4      Π_i = ∅;
5  while I ≠ ∅ do
6      select a node i ∈ I;
7      for (i, j) ∈ A do
8          H_ij :← ∅;
9          for L_p = (T_p^load, T_p^tMin, T_p^tMax, T_p^rtMax, a_p, V_p^V) ∈ Π_i;
10         do
11             if V_p^j = 0 or j ∉ NG_i then
12                 if Extend(i, j, L_p)=true then
13                     H_ij :← H_ij ∪ L_p;
14         Π_j ← Dominance(Π_j ∪ H_ij);
15         if Π_j has changed and j ∉ I then
16             I ← I ∪ {j};
17     I ← I\{i}
18 Π_0 = Dominance(Π_0);
19 if ∃ elementary routes with negative reduced costs at the depot then
20     update Ω_R and solve RLPMP;
21 else if ∃ non-elementary routes with negative reduced costs at the depot then
22     select the customer k with the highest multiplicity and NG_h ← {k}, h ∈ V\{k} ;
```

We develop the heuristic labeling algorithm based on the *ng-route* algorithm to solve the pricing subproblem. The preliminary experiments show that removing $\mathbf{V}_p^V$ from the label enhances the performance of the CG algorithm.

Since the relaxation does not guarantee an optimal solution, we propose an enhanced CG procedure that uses the Heuristic Labeling Algorithm (HLA) to effectively determine a good upper bound to the RLPMP. In other words, the *Two-Phase* Algorithm (TPA) is developed for the exact BPC by benefiting from the fast execution of the HLA. In the first stage, the HLA produces routes that feed the RLPMP until no more columns with negative reduced cost can be found. Then, in the second stage,

a CG procedure is applied using the *ng-route* algorithm. The second stage of the TPA certifies that the obtained solution is optimal.

Algorithm 2.2: Extend(i, j, L_p)

```
Calculate T_j^load, T_j^tMin, T_j^tMax and T_j^rtMax using equations (2.20)-(2.33);
if T_j^tMin ≤ l_j & T_p^tMin ≤ T_j^tMax & T_j^rtMax ≤ gQ then
    if j ∈ V then
        a_p ← a_p + 1;
        V_p^j ← V_p^j + 1;
    L_p = (T_j^load, T_j^tMin , T_j^tMax, T_j^rtMax, a_p, V_p^V) ;
else
    a_p ← a_p + 1 ;
    V_p^j ← V_p^j + 1 ;
```

2.3.4.3. Bidirectional Search

In the bidirectional search mechanism, we employ both the forward labeling procedure described in Section 2.3.1.1 and a backward labeling algorithm (see Righini and Salani, 2006 for the details about backward labeling). The backward search starts from the arrival depot, and finishes at the departure depot. Each path starts at time l_0, and the time is decreased as more nodes are visited. Let $L_p = (W_i^{load}, W_i^{tMin}, W_i^{tMax}, W_i^{rtMax}, w_p, \mathbf{B}_p^V)$ be the label of the backward path p. The definitions of W_i^{load}, w_p and $\mathbf{B}_p^V$ are similar to those given for T_i^{load}, a_p and $\mathbf{V}_p^V$, respectively (see Section 2.3.1.1). Moreover, W_i^{tMin}, W_i^{tMax} and W_i^{rtMax} are used to manage the arrival time and the battery capacity as before. The related formulations are presented as follows:

$$W_j^{tMin} = \begin{cases} \min\left(l_j, W_i^{tMin} - t_{ij} - s_j\right), & \text{if } |S_p| = 0 \\ \min\left(l_j, W_i^{tMin} - t_{ij} - s_j\right) - R_{ij}, & \text{otherwise} \end{cases} \tag{2.31}$$

$$W_j^{tMax} = \begin{cases} \max\left(e_j, \min(l_j, W_i^{tMin} - W_i^{rtMax} - t_{ij} - s_j)\right), & \text{if } i \in F \\ \max\left(e_j, \min(l_j, W_i^{tMax} - t_{ij} - s_j)\right) & \text{otherwise} \end{cases} \tag{2.32}$$

$$W_j^{rtMax} = \begin{cases} W_i^{rtMax} + gd_{ij}, & \text{if } |S_p| = 0 \\ \min\left(gQ, \max(0, W_i^{rtMax} - S_{ij} + gd_{ij})\right) & \text{otherwise} \end{cases} \tag{2.33}$$

where

$$S_{ij} = \begin{cases} \max\left(0, \min(W_i^{tMin} - t_{ij} - s_j - l_j, W_i^{rtMax})\right) & \text{if } i \in F \\ \max\left(0, \min(W_i^{tMin} - t_{ij} - s_j - l_j, W_i^{tMin} - W_i^{tMax})\right) & \text{otherwise} \end{cases}$$

$$R_{ij} = \max\left(0, \max(0, W_i^{rtMax} - S_{ij}) + gd_{ij} - gQ\right)$$

Since the route time is decreased each time the path is extended to a node, a change is required in the treatment of the service times. More specifically, to compute the arrival time at node j, the service time of the customer j needs to be subtracted.

Forward paths and backward paths are extended until a predefined time $M \in (e_0, l_0)$. The exploitation stops when both T_i^{tMin} and W_i^{tMin}, are less than or equal to M for forward and backward labeling, respectively. A forward label and a backward label at a node are combined if the conditions (2.34)-(2.37) hold.

$$T_j^{load} + W_j^{load} - q_j \leq C, \tag{2.34}$$

$$V_p^v + B_p^v \leq 1 \ \ \forall v \in V_{0,n+1} \backslash \{j\}, \tag{2.35}$$

$$T_i^{tMin} + Z_i \leq W_i^{tMin}, \tag{2.36}$$

$$Z_i \leq (T_i^{tMax} - T_i^{tMin}) + (W_i^{tMin} - W_i^{tMax}) \tag{2.37}$$

where $Z_i = \max(0, T_i^{rtMax} - W_i^{rtMax} - gQ)$ denotes the time required at a station for recharging the minimum amount of energy to ensure that the battery SoC is never negative on the route (Desaulniers et al., 2016). Condition (2.34) indicates that the total load of the vehicle does not exceed its capacity. Condition (2.35) ensures that customers are not visited more than once. Nevertheless, this condition may be relaxed for the *ng-route* algorithm and B_p^v can take values greater than one as well. Condition (2.36) guarantees that the time window of node i is respected. Condition (2.37) ensures that the minimum required recharging time does not exceed the available route time. Note that this condition is not checked if i represents a recharging station.

Single Charging Problem. For the EVRPTW with single charging, the backward labeling procedure needs to limit the total number of stations visited en route to one as in forward labeling. Moreover, to obtain a feasible combination of a forward and a backward path the total number of stations in the completed path should be less than or equal to one.

2.3.4.4. Reducing the Size of the Search Tree

Let F_{ij} be the set of nondominated stations for node pair (i, j) and $s^* = \arg\min_{s \in F}\{d_{is} + d_{sj}\}$. For all $s \in F \backslash \{s^*\}$ if $d_{is} \geq d_{is^*}$ and $d_{sj} \geq d_{s^*j}$, then s is dominated by s^* and removed from F_{ij} (see Bruglieri et al., 2016). Consequently, we can make the following observation. In the optimal solution, recharging stations between each node pair (i, j) belong to the set F_{ij}.

Figure 2.1. illustrates an example involving four stations that the vehicle can visit during its trip from customer i to customer j. Among the four stations, $d_{is_2} + d_{s_2j} = 15$ is the minimum. Hence, s_4 is dominated since $d_{is_4} = 11 > 10 = d_{is_2}$ and $d_{s_4j} = 11 > 5 = d_{s_2j}$.

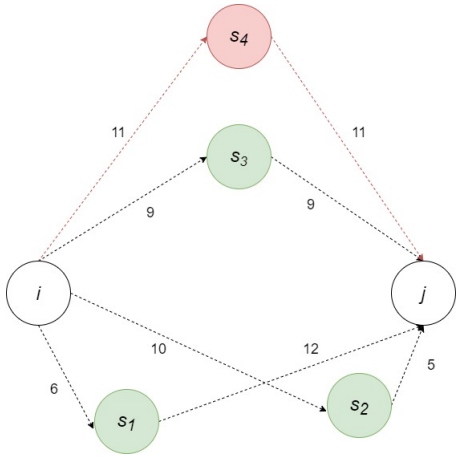

Figure 2.1. An illustrative example for station dominance (Keskin and Çatay, 2018)

In the route generation algorithms developed for the EVRPs (Desaulniers et al., 2016; Taş, 2021), a path arriving at customer i can be extended to an unvisited customer j, or to a recharging station s, or to the depot. If F includes too many stations, the search tree can be massive. As a remedy, using Observation 2.3.4.4, we introduce the Augmented Node (AN) approach which enables a more effective search on the tree. When a vehicle is at customer i, it will then visit a customer node j or an augmented node, j^s, which is a combination of node j and a non-dominated station s.

In this way, EVs always travel from one customer to another since visits to stations are performed implicitly. Since many stations are remotely located, F_{ij} can be significantly smaller than F. Thus, the number of nodes that the algorithm searches through is expected to decrease significantly.

In both TPA and HCG, a path is extended to node j^s by augmenting label calculations for station s and node j. The extension of the label L_p to augmented node j^s requires to be performed first for station s and then for customer j using equations (2.20)-(2.33).

2.3.4.5. Bounding Procedure

A number of methods have commonly been used in the VRPTW literature to fathom suboptimal paths. Bounding methods aim at producing lower bounds on the reduced costs based on the consumption of a single resource where the utilization of all other resources are often ignored. The bounding method suggested by Baldacci et al. (2011) generates a bound matrix $B(r,t)$ which is obtained by finding lower bounds for every node i with respect to discrete values of time consumption. The procedure relaxes the capacity constraint and assumes that vehicles have an unlimited freight capacity.

Algorithm 2.3: Bounding Procedure

```
1  τ ← l_0;
2  while τ ≥ e_0 do
3      τ ← τ − δ;
4      for i ∈ N do
5          p = {};
6          c̄_p = 0;
7          T_i^load = 0;
8          T_i^tMin = τ;
9          T_i^tMax = τ;
10         T_i^rtMax = 0;
11         Explore(p, c̄_p, T_i^load, T_i^tMin, T_i^tMax, T_i^rtMax) ;
12         if ∄ any routes at the depot then
13             B(i, τ) = ∞;
14         else
15             B(i, τ) = c̄_p;
```

The method represented by Algorithm 3.3 is employed as follows. Given that l_0 is the latest possible arrival time at the depot and δ is a nonnegative time step, the pricing subproblem is solved for every node $i \in N$ and the search starts at $t = l_0 - \delta$. In the first iteration, since only δ units of time is available, the resulting number of feasible solutions is small. These solutions can be rapidly found using a labeling

algorithm represented by the `Explore()` procedure in Line 11 of the Algorithm 3.3. For this procedure, we employ HLA since this is the fastest algorithm that we propose to generate routes. The lower bound for node i is the minimum reduced cost of any path at node i which uses maximum δ units of time. In the next iteration, the available time consumption is increased by δ, and the lower bounds on reduced cost for every node are found with respect to the starting time $l_0 - 2\delta$. Although the size of the feasible region is increased, the search can be completed in short time since the lower bounds found so far can be used to fathom unpromising paths. This procedure is repeated until the starting time reaches to the lower bound of the time window of the depot, e_0. If the procedure `Explore()` cannot find any routes with negative reduced costs, then the algorithm stops.

A bound check function is implemented in both stages of TPA. The function fathoms any path if $\bar{c}_p + B(i, t(p)) > \bar{c}_p*$, where $\bar{c}_p*$ is the global bound continuously updated with the minimum reduced cost of a complete route.

2.3.4.6. Generating Upper Bounds with Integral Master Problem

The performance of the BPC procedure strongly depends on the quality of the bounds. Hence, it suffers deeply if the bounds are not updated for a considerable number of iterations. Especially, the depth-first search strategy may cause a delay in updating lower bounds since the algorithm may generate several new branches without obtaining any integral solution. In such cases, the lack of good upper bounds increases the number of required tree nodes, and hence the computation time. We address this issue and solve the master problem (2.15)-(2.17) when the root node of the BPC tree ends with a fractional solution. The Integral Master Problem (IMP) generates an integer feasible solution which provides a good upper bound to the problem using all the routes generated at the root node. The IMP reduces the computational effort, since the relative integrality gap shrinks faster with a good upper bound.

2.3.5. Heuristic Column Generator

When the second component of TPA is discarded, the resulting algorithm corresponds to a heuristic algorithm based on a CG scheme. As mentioned in Section

4.2.3.2, the HLA is obtained by removing the vectorial label components of forward and backward labeling algorithms, $\mathbf{V}_p^V$ and $\mathbf{B}_p^V$ both from the label calculations and the dominance procedure of the *ng-route* algorithm. The computation time required to execute the HCG is certainly less compared to the TPA because of the relaxation in the labeling algorithm and the elimination of the CG procedure including the *ng-route* algorithm. On the other hand, refining in the dominance procedure may also cause the elimination of the optimal routes. Therefore, the HLA does not provide lower bounds but it generates good upper bounds. Similar to the TPA, the heuristic procedure is enhanced with the employment of all acceleration methods and the subset-row inequalities.

2.4. Computational Study

We conducted our computational experiments on the EVRPTW data provided by Desaulniers et al. (2016) generalized based on the well-known VRPTW benchmark dataset of Solomon (1987). There are three groups of instances categorized according to the geographical positions of customers: clustered (C), randomly located (R), and randomly clustered (RC). Moreover, each set includes two different groups with respect to the width of the time windows and the length of planning horizon: 100-series (tight time windows), and 200-series (wide time windows). In addition, Desaulniers et al. (2016) introduce instances with 25 and 50 customers by reducing the size of the original data involving 100 customers and 21 recharging stations. We skipped C2, R2 and RC2 instances since the time window constraints can easily be satisfied and their effect on routing and recharging decisions is minor (Desaulniers et al., 2016; Keskin and Çatay, 2018), and focused only on the C1, R1 and RC1 instances.

Our algorithms were implemented on JAVA (using IntelliJ Idea, 2020) and linear programming models were solved by IBM ILOG CPLEX 12.9. (IBM, 2020) using a computer with 3.2 GHz Intel i7-8700 processor and 32 GB RAM running on Windows 10 Pro operating system. For consistency, we followed the approach of Desaulniers et al. (2016) when performing the computational analysis including the rounding methodology that they applied to the parameter values. We limited the execution time to one hour for each instance.

In our computational analysis, we set the cuts related parameters $t^{SRC} = 0.2$

and $n_{max}^{SRC} = 10$ (Desaulniers et al., 2016). To tune the remaining parameters, we performed preliminary experiments on a subset of instances by considering $n^{col} = \{25, 50, 75, 100, 150\}$, $n^{iter} = \{1, 2, 3, 4, 5\}$ and $n^{rt} = \{10, 100, 500, 1000\}$, and determined $n^{col} = 50$, $n^{iter} = 2$, and $n^{rt} = 100$ for the monodirectional algorithms. The ICP parameters remain the same for bidirectional methods. However, since forward and backward search mechanisms stop before the latest possible arrival time to the depot is reached, the number of routes at the depot decreases significantly in the bidirectional algorithms. Based on our preliminary tests conducted for $n^{rt} = \{1, 5, 10, 20\}$ we set $n^{rt} = 5$. In the bidirectional algorithm, forward and backward paths are stopped at $M = 0.5(l_0 - e_0)$ for the exact labeling. For the heuristic labeling, we performed experiments on a subset of instances by testing $M = \{(l_0 - e_0), 0.95(l_0 - e_0), 0.90(l_0 - e_0), 0.75(l_0 - e_0)\}$ and set M to $0.90(l_0 - e_0)$.

Additional computational experiments are performed with bidirectional two-phase algorithm to measure the efficiency of the *ng-route* algorithm and bounding method by considering the following categories: (i) neither ng-route nor bounding, (ii) only ng-route, (iii) only bounding, and (iv) both ng-route and bounding by using a subset of six instances involving 50 customers. The experiment setting including both methods has the shortest average computational time and all instances are solved. On the other hand, using the setting in which none of the methods are included, show that only 4 instances could be solved and the average computational time is the largest among all. This study also shows that the *ng-route* improves the performance of the algorithm more compared to the bounding method for the selected subset of instances.

Table 2.2. provides a summary of the numerical results obtained by using the TPA and HCG. SP and MP in the first column refer to the single-partial and multiple-partial recharge cases, respectively. Columns #Opt, t(s) and Δ(%) under TPA indicate the number of instances solved to optimality, the average computational time in seconds and the relative integrality gap, respectively. The relative integrality gap is the gap between the best upper (z^*) and the best lower (z^{lb}) bounds and is calculated as $100\frac{z^* - z^{lb}}{z^*}$. Columns #Solved and Δ_{TPA} (%) under HCG report the number of instances solved and the average difference between the objective functions of HCG and TPA, respectively. Δ_{TPA} is calculated as $100\frac{z^h - z^*}{z^h}$, where z^h is the objective function value obtained by HCG. For a fair comparison Δ_{TPA} is computed only over the instances solved by both algorithms.

The results in Table 2.2. show that both TPA and HCG can easily solve all 25-customer MP and SP instances with bidirectional as well as monodirectional search. However, we see that the bidirectional search does not enhance the run time. This

may be due to the fact that it requires storing substantially more paths. In addition, the time spent for combining forward and backward paths increases exponentially as the number of partial routes increases.

In the case of 50-customer MP instances, the performances of the HCG and TPA are similar in terms of the number of problems solved. On the other hand, both methods benefit from the bidirectional search in solving additional problems and the HCG is significantly faster than the TPA, on average. In 100-customer MP instances, HCG outperforms TPA by solving four and eight more instances with monodirectional and bidirectional search, respectively.

The results are similar for the SP instances: the performances of the two methods with both monodirectional and bidirectional labeling are comparable for the 50-customer instances with respect to the number of problems solved while, on average, HCG requires significantly less effort. Regarding 100-customer SP instances, HCG again outperforms TPA by solving five more problems using monodirectional search and eight more using bidirectional search. Overall, we see that the average run time of the HCG is considerably shorter than that of the TPA. Furthermore, bidirectional search improves the efficiency of both methods in solving large-size instances. It makes a significant contribution to the performance of the HCG in particular, allowing it to solve seven additional problems in both MP and SP variants.

Comparing the results for the two problem variants, we observe that the algorithms exhibit a similar performance in solving both, even though only one recharge is allowed en-route in the SP case. The relaxation of path labels allows the dominance procedure of the HCG to be shorter than that of the TPA, which reduces the average computation time approximately by 50%. Moreover, this relaxation provides extremely good bounds for the subproblem.

In sum, we see that out of 174 test instances, the TPA and HCG with monodirectional search solve 128 and 136 instances, respectively. TPA can solve eight more instances by using bidirectional labeling, whereas HCG solves 16 more. The algorithm is completed at the root node for 551 solutions obtained. This is achieved by the implementation of subset-row inequalities with executing integral master problem. The detailed results are provided in Appendix A.

Table 2.3. presents the results for the instances for which either the best-known solution (BKS) is improved or a solution is introduced to the literature for the first time. The third column reports BKS values provided to the literature by Desaulniers et al. (2016). The best solution value obtained by our algorithms are provided in the

fourth column. The fifth column indicates the name of the algorithm providing that solution. In order to better assess the quality of the upper bounds found by TPA and HCG, we executed TPA with a time limit of five days in an attempt to determine good lower bounds. So, Δ (%) in the table reports the relative integrality gaps calculated using these lower bounds. The last column reports the computational time in seconds.

Table 2.2. Summary of results

			Monodirectional Search						Bidirectional Search					
			TPA			HCG			TPA			HCG		
Problem Variant	n	Type	#solved	t (s)	Δ (%)	#solved	t (s)	Δ_{TPA} (%)	#solved	t (s)	Δ (%)	#solved	t (s)	Δ_{TPA} (%)
MP	25	All	29/29	14.7	0.94	29/29	5.2	0.10	29/29	17.7	0.84	29/29	4.1	0.06
		C-set	9/9	18.8	0.46	9/9	5.8	0.18	9/9	36.2	0.43	9/9	3.9	0.08
		RC-set	8/8	6.1	1.12	8/8	3.0	0.02	8/8	5.1	0.94	8/8	2.5	0.07
		R-set	12/12	17.4	1.19	12/12	6.1	0.09	12/12	12.1	1.09	12/12	5.4	0.04
	50	All	26/29	342.3	1.73	27/29	136.5	0.02	27/29	210.9	1.57	29/29	171.6	0.25
		C-set	8/9	162.8	0.87	9/9	230.4	0.01	9/9	274.7	0.99	9/9	68.6	-0.03
		RC-set	7/8	234.5	2.62	7/8	65.5	0.00	7/8	115.2	2.60	8/8	431.6	0.39
		R-set	11/12	541.3	1.78	11/12	104.9	0.04	11/12	220.5	1.37	12/12	75.5	0.41
	100	All	8/29	950.3	2.12	12/29	784.4	0.00	11/29	831.1	2.52	19/29	904.0	0.19
		C-set	3/9	843.2	3.01	3/9	804.7	0.00	3/9	961.0	3.05	5/9	760.3	0.14
		RC-set	2/8	1319.5	2.55	3/8	263.4	0.00	3/8	901.6	3.49	3/8	235.0	0.39
		R-set	3/12	811.2	1.09	6/12	1034.8	0.00	5/12	630.7	1.03	11/12	1151.8	0.11
SP	25	All	29/29	7.1	0.79	29/29	2.5	0.00	29/29	7.3	0.84	29/29	2.1	0.00
		C-set	9/9	9.0	0.52	9/9	2.6	0.00	9/9	9.2	0.53	9/9	1.7	0.00
		RC-set	8/8	3.2	0.87	8/8	1.7	0.00	8/8	3.1	0.77	8/8	1.7	0.00
		R-set	12/12	8.2	0.94	12/12	3.0	0.00	12/12	8.8	1.11	12/12	2.8	0.00
	50	All	26/29	211.0	1.99	27/29	44.8	0.46	27/29	317.7	2.09	27/29	35.6	0.11
		C-set	8/9	60.3	1.42	9/9	46.9	0.83	9/9	212.5	1.27	9/9	27.7	0.24
		RC-set	6/8	150.0	1.73	6/8	39.2	0.83	6/8	128.3	1.85	6/8	35.6	0.06
		R-set	12/12	342.0	2.50	12/12	46.1	0.02	12/12	491.3	2.83	12/12	41.5	0.03
	100	All	10/29	857.8	1.66	15/29	716.2	0.24	14/29	1128.5	1.90	22/29	729.7	0.14
		C-set	5/9	768.3	1.26	6/9	683.1	0.00	6/9	1120.0	1.27	7/9	834.5	0.10
		RC-set	1/8	1877.5	4.85	2/8	578.3	2.43	3/8	1756.0	3.13	4/8	803.9	0.34
		R-set	4/12	714.7	1.38	7/12	783.9	0.00	5/12	762.2	1.91	11/12	636.0	0.21

We provide 21 new solutions in Table 2.3. , 13 in the MP-type and eight in the SP-type instances. Out of 21 solutions, seven were obtained using monodirectional search (indicated with [a]). Furthermore, 16 new solutions were achieved by using HCG, which validates the superior performance of the proposed heuristic approach. Note that the solutions for instances C106 and C107 were improved while testing the branching rules in TPA (indicated with [b]).

Table 2.3. New and improved solutions

				this study			
Problem Variant	n	Instance	BKS	Bound	Method	Δ (%)	t (s)
MP	50	C103	n/a	63641	TPA	3.05	626.9
		RC107	n/a	78368[a,c]	HCG	4.11	59.1
		R104	n/a	62199[c]	HCG	3.56	185.6
	100	C102	n/a	101542	HCG	4.66	1461.5
		C109	n/a	93688	HCG	4.59	1257.3
		RC101	n/a	163949	TPA	4.39	303.9
		R104	n/a	106742	HCG	3.97	2670.2
		R106	n/a	120760[a]	HCG	0.45	2446.0
		R107	n/a	112227	HCG	3.04	2327.3
		R108	n/a	100652	HCG	3.75	2169.5
		R109	n/a	118519	TPA	3.08	1921.2
		R110	n/a	109386	HCG	4.78	1474.5
		R111	n/a	109820	HCG	4.71	1451.0
SP	50	R104	n/a	62561[a,c]	HCG	3.74	165.1
		R112	n/a	66054	HCG	4.85	84.7
	100	C106	102744	102467[a,b]	TPA	0.35	3239.8
		C107	103040	102665[a,b]	TPA	1.75	2689.8
		RC106	n/a	140885	TPA	4.85	1877.5
		R105	n/a	137557[a,c]	HCG	2.67	50.2
		R107	n/a	117628	HCG	3.77	1167.8
		R109	n/a	126350	TPA	3.25	1056.4
		R110	n/a	111691	HCG	4.77	668.5
		R111	n/a	112729[a]	HCG	3.05	1294.1

[a] Obtained by using monodirectional search

[b] Obtained when testing the branching rules

[c] Obtained with TPA as well.

2.5. Conclusion

This chapter presents exact and heuristic algorithms based on BPC method to solve the EVRPTW. For both algorithms, a column generation method including a pricing subproblem and a restricted linear programming master problem is employed. The pricing subproblems are solved by labeling algorithms which are developed by modifying the well-known *ng-route* algorithm. On the other hand, the heuristic method is obtained by relaxing the *ng-route* algorithm whereas the CG procedure of the exact algorithm contains two steps: a heuristic labeling algorithm followed by the *ng-route* algorithm.

The performance of the algorithms is improved by using several further acceleration techniques including the bidirectional search improving the performance of labeling algorithms, Intermediate Column Pool (ICP) preserving a number of routes with non-negative reduced cost to be used in later iterations, Augmented Node (AN) method enabling the elimination of many inefficient searches through the network, the bounding method eliminating partial paths using the generated lower bounds on reduced costs, and Integral Master Problem (IMP) helping the algorithms produce a good upper bound.

The algorithms are evaluated by using a well-known data set which includes instances with up to 100 customers. Considering only the instance solved by both algorithms, the computational time of the heuristic algorithm is at least 55% less than the exact algorithm for each experiment setting. Moreover, the average of relative distance of the solutions obtained by using HCG and TPA is no more than 1%. Lastly, we introduce 21 new solutions to the literature that can be used in future research and improve solutions of two instances that have already been provided in the literature.

The efficiency of the heuristic algorithm is shown by the computational tests. This method simply outperforms the TPA especially for large-size instances. Moreover, for small- and medium-size instances the heuristic method provides similar results compared to the exact algorithm. Therefore, the heuristic algorithm can be used to solve the real-life problems which usually include a large number of customers.

The methods proposed in this study can be further improved by implementing the network reduction techniques to solve larger problem instances. Further research may also focus on solving extensions of the EVRP such as the versions considering multiple recharge technologies, flexible deliveries with alternative locations and time

windows, and uncertainties related to service and recharging times.

3. IMPLEMENTATION OF PULSE ALGORITHM FOR SOLVING THE ELECTRIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

In this chapter, we solve the EVRPTW by implementing the Pulse algorithm for the pricing subproblem of a Branch-and-Price (BP) procedure. The Pulse algorithm is suggested by Lozano et al. (2015) to solve the VRPTW within a Column Generation (CG) procedure. The algorithm simplifies the difficulties of the classical labeling algorithms such as label storage and dominance. Its depth-first search structure prevents the storage of an excessive number of labels at the same time. The Pulse algorithm consists of several strategies which fathom suboptimal paths. One of these strategies is a bounding method which creates a bound matrix using lower bounds on the reduced costs found for each node and discrete values of resource consumption. The bound matrix is beneficial to eliminate unfavorable paths and thereby speeding up the process. In addition, another method called rollback pruning, often used in labeling algorithms (Feillet et al., 2004; Kohl et al., 1999), simplifies the dominance procedure by evaluating whether the last node included in the path should be removed. The construction of this strategy allows the disposal of most of the dominance rules.

Lozano et al. (2015) present a computational study to evaluate the performance of the Pulse algorithm by solving the root node of the BP tree on the data set introduced by Solomon (1987). The results obtained by the Linear Programming (LP) relaxation are compared to those provided by Baldacci et al. (2011). Furthermore, three more studies using the Pulse algorithm to solve a variety of Constrained Shortest Path problems (CSP) are described as follows.

Thomas et al. (2019) address the family of resource-constrained shortest path problems and present three algorithms including a bidirectional Dijkstra's method and the Pulse algorithm to solve them. The study provides a literature review on the heuristic bidirectional search approaches and benefits from one of these methods to improve the performance of the bidirectional Dijkstra's method. In the heuristic

pricing, the authors modify the resource bounding method suggested by Righini and Salani (2006) in which extension of a path is terminated if more than half of a resource referred to as "critical" is consumed. The authors estimate the shortest path distances approximately while employing the resource bounding method. The article is completed with an extensive computational study evaluating the performances of several methods from the literature using a real-life data set. It is stated that the Pulse algorithm obtains shorter average computation time than the bidirectional Dijkstra's method, but it runs out of memory for many instances.

Cabrera et al. (2020) solve the CSP and the multi-activity shift scheduling problem which deals with scheduling staff who are responsible for different work activities during different periods. A BP algorithm, benefiting from a bidirectional Pulse algorithm to generate routes, is proposed to solve the problems. The method includes a Pulse-based heuristic that improves the CG algorithm. The authors provide a computational study over large real-road networks with up to six million nodes and 15 million arcs.

Li et al. (2020) provide a model for the plug-in hybrid electric vehicles which minimizes the energy consumption of EVs. The authors propose a BP algorithm to solve the CSP by employing the Pulse algorithm to generate the route with the minimum reduced cost. The performance of the algorithm is evaluated using a simulation model of the Ann Arbor city in SUMO. The experiment results provide that the total energy consumption of EVs is reduced by 7% to 14%.

We modify the labels reported by Lozano et al. (2015), to cope with battery energy consumption and recharging requirements of EVs. Relevant dominance rules are included in the rollback pruning method. Furthermore, the BP procedure with the Pulse algorithm is improved using several acceleration techniques including the Integral Master Problem (IMP), Intermediate Column Pool (ICP), and the Augmented Node (AN) methods (see Section 2.3.4). We employ the AN method which reduces the problem network by eliminating the dominated stations dynamically and explores the search tree by traveling from one customer to another either directly or via a station. IMP is applied to obtain a strong upper bound for the problem at the root node of the BP tree. ICP participates in increasing the performance of the algorithm by storing routes with nonnegative reduced costs to be used in later CG iterations without causing an important amount of time. Additionally, each iteration of the Pulse algorithm is prematurely terminated when the number of complete routes with negative reduced cost reaches a threshold value. For the branching, we provide three rules that are branching (i) on the total number of vehicles, (ii) on the total number of recharges, and (iii) on the total flow value of an arc. We dis-

card branching on the recharges in a station (see Section 2.3.2), since it significantly increases the information to be preserved at each branch.

In the computational analysis, we evaluate the proposed procedures by using the benchmark instances provided by Desaulniers et al. (2016) including up to 100 customers and 21 stations. We compare the results with those obtained in Chapter 2 and provide solutions for several instances that have not been solved before.

The rest of this chapter is organized as follows. In Section 3.1, we provide the details of the Pulse algorithm. Section 3.2 describes the experimental design and presents the computational results. In the computational study, first, we provide the results of the preliminary tests performed to determine the best ICP parameters and the threshold value on the number of routes to terminate the algorithm prematurely. Then, using the determined parameters two versions of the Pulse algorithm, with and without AN, are evaluated and the results are compared to those provided in Section 2.4. Finally, the summary and concluding remarks are provided in Section 3.3.

3.1. Solution Methodology

First, we apply the preprocessing and the generating initial solutions using the methods described in Section 2.3. Then, the BP procedure begins by solving the root node of the BP tree. The CG algorithm is employed to solve each branch. The master problem of the CG corresponds to the set partitioning formulation represented as the formulation (2.15)-(2.17). The pricing subproblem of the CG is solved with the Pulse Algorithm. Additionally, several acceleration methods are implemented to improve the performance of the CG. If the CG algorithm ends with a fractional solution, the branching procedure finds two new branches by applying the first attainable branching rule among the three rules mentioned in the earlier section.

The Pulse algorithm explores the network and generates partial paths until they reach the depot or are eliminated by a pruning strategy. It consists of two general steps: first a bounding procedure finds lower bounds on the reduced cost given an amount of resource consumption for each node, and then the routes with negative reduced costs are recursively generated based on an implicit enumeration of the solution space.

The exploration stage works similar to the labeling algorithms. It begins at the depot node, travels throughout the outgoing arcs of each visited node using a depth-first search structure and stores the resulting partial path p at each node i with its the reduced cost $\bar{c}_p$ and the cumulative consumption values for each resource. The first resource q_p is the freight load of path p. In Lozano et al. (2015), the second label developed for the VRPTW is the total time consumption of path p. However, a single label is inadequate to determine the cumulative time for the problem including EVs. Instead, we modify the algorithm by including three time-related resources, T_i^{tMin}, T_i^{tMax}, and T_i^{rtMax}, which denote the arrival time to node i considering the battery SoC feasibility, the arrival time to node i considering the time window feasibility, and the required time for recharging the battery fully, respectively, as in the generic labelling algorithm described in Section 2.3.1.1.

Since our problem allows partial recharges, the optimal amount of energy to be recharged at a station is unknown before EVs arrive at the depot. This issue is resolved by using two different labels for the cumulative time consumption of the path p, regarding the minimum and the maximum possible recharging time. The minimum possible recharging time is considered for maintaining battery feasibility. On the other hand, the maximum recharging time is determined concerning the battery capacity and upper limit of the time window. If there is no station along path p, the arrival time is calculated as the summation of the travel duration and the service times at the customers, thus the values of the two time resources, T_i^{tMin} and T_i^{tMax}, are equal. The difference between these labels is explained as follows. The resource T_i^{tMin} is the arrival time at node i considering the minimum amount of recharging time that sustains the battery feasibility if path p visits a station before i. On the other hand, T_i^{tMax} provides the arrival time at node i considering the maximum amount of recharging time with respect to the service closing time of i, if recharging occurs at the prior nodes. Moreover, the difference between T_i^{tMin} and T_i^{tMax}, $T_i^{tMax} - T_i^{tMin}$, shows the additional time that can be used for recharging besides the minimum required recharging time and is always greater than or equal to zero for a feasible path. The calculation of these two labels still requires the knowledge of the battery SoC at each node. The resource T_i^{rtMax} refers to the time for recharging the battery fully assuming that if a station is visited along the path, then the minimum amount of energy, maintaining battery feasibility, is loaded into the vehicle.

Algorithm 3.1: Pulse Algorithm

1 Start at node 0, $q_p = 0$, $\bar{c}_p = 0$, $T_i^{rtMax} = 0$, $T_i^{tMin} = 0$, $T_i^{tMax} = 0$;
2 $bound(\delta)$;
3 $pulse(\bar{c}_p, q_p, T_i^{tMax}, T_i^{tMin}, T_i^{tMax})$;
4 Return ϕ

Algorithm 3.1 shows the general structure of the Pulse algorithm which starts with the initialization. The algorithm then generates a bound matrix containing lower bounds on the reduced costs for each node and the time step over the planning time horizon, the length of which is δ. Next, the *pulse* procedure is applied which is explained in Algorithm 3.2. Lastly, the algorithm ends with ϕ, a set of negative reduced cost columns.

Algorithm 3.2: *pulse* Procedure

1 **if** $Feasible(\bar{c}_p, q_p, T_i^{rtMax}, T_i^{tMin}, T_i^{tMax}) = true$ **then**
2 **if** $checkBound(\bar{c}_p, q_p, T_i^{rtMax}, T_i^{tMin}, T_i^{tMax}) = false$ **then**
3 **if** $rollback(\bar{c}_p, q_p, T_i^{rtMax}, T_i^{tMin}, T_i^{tMax}) = false$ **then**
4 $p \leftarrow p \cup \{j\}$;
5 $q_p \leftarrow q_p + q_j$;
6 $\bar{c_p} \leftarrow \bar{c_p} + \bar{c_{ij}}$;
7 $T_i^{tMin} \leftarrow LCE(T_i^{tMin}, T_i^{tMax}, T_i^{rtMax}, t_{ij}, s_i, h_{ij})$;
8 $T_i^{tMax} \leftarrow LCE(T_i^{tMin}, T_i^{tMax}, T_i^{rtMax} t_{ij}, s_i)$;
9 $T_i^{rtMax} \leftarrow LCE(T_i^{rtMax}, h_{ij})$;
10 $pulse(\bar{c}_p, q_p, T_i^{rtMax}, T_i^{tMin}, T_i^{tMax})$

In Algorithm 3.2, the steps of *pulse* procedure are provided explicitly. First, the algorithm checks whether path p is feasible. Then, the bound check is completed. Next, the rollback pruning function reevaluates the latest node choice of the path. If path p passes all three pruning strategies, then it is extended to node j. The details of each procedure are explained as follows.

The Pulse algorithm includes not only the propagation to new nodes but also three pruning strategies. The first one is infeasibility pruning which eliminates a path if it is infeasible. A path is infeasible in following situations: (i) a customer is visited more than once, (ii) the arrival time considering battery feasibility is later than the service closing time of customer i ($T_i^{tMin} > l_i$), (iii) the arrival time considering battery feasibility is later than the arrival time considering time window feasibility ($T_i^{tMin} > T_i^{tMax}$), (iv) the cumulative load exceeds load capacity of the vehicle

C ($q_p > C$), and (v) the maximum required recharging time is greater than the recharging time for an empty battery ($T_i^{rtMax} > gQ$).

The second pruning strategy is a bound check procedure that eliminates paths if the summation of their reduced cost and the lower bound is greater than a continuously updated global bound on the reduced cost of a route, $\bar{c}_p^*$. The lower bounds are generated iteratively through the bounding procedure and used to eliminate the suboptimal paths. The reduced cost achieved by a path that uses $t(p)\delta = \tau$ amount of time is a lower bound for the reduced cost of that path with $\tau+\epsilon$ time consumption where $\epsilon \geq 0$ and $t(p)$ is the time step. As an example, let us consider a problem with the total length of the day as 100 and a time step defined as 10. If T_i^{tMin} is 75 units of time, then the corresponding time step on the bound matrix is 70 units of time. Note that under this bounding scheme, time step $t(p)$ is the lower closest available value in the bound matrix to the arrival time of path p; thus, $\tau \leq T_i^{tMin} \leq T_i^{tMax}$. Since T_i^{tMin} is the earliest arrival time of path p that is closest to the step length τ, it is used to calculate the time step $t(p)$ and p is pruned if $\bar{c}_p + B(i, t(p)) > \bar{c}_p^*$.

Let path p_1 be a partial path at node j which travels through arcs (i, k) and (k, j), respectively. For path p_1, the rollback pruning strategy reevaluates the decision of visiting node k before node j by comparing its resource consumption to the amount of that consumed by an alternative path p_2 which visits the same nodes in the same order as in path p_1 until node i and then travels to node j directly. Dominance rules described by Feillet et al. (2004) state that path p_1 dominates path p_2 if and only if the following conditions are held and at least one of them is strictly satisfied:

$$\bar{c}_{p_1} \leq \bar{c}_{p_2} \tag{3.1}$$
$$a_{p_1}^V \leq a_{p_2}^V \tag{3.2}$$
$$V_{p_1}^j \leq V_{p_2}^j \text{ for all } j \in V \tag{3.3}$$
$$q_{p_1} \leq q_{p_2} \tag{3.4}$$
$$T_{p_1}^{tMin} \leq T_{p_2}^{tMin} \tag{3.5}$$
$$T_{p_1}^{rtMax} - (T_{p_1}^{tMax} - T_{p_1}^{tMin}) \leq T_{p_2}^{rtMax} - (T_{p_2}^{tMax} - T_{p_2}^{tMin}) \tag{3.6}$$
$$T_{p_1}^{rtMax} + T_{p_1}^{tMin} \leq T_{p_2}^{rtMax} + T_{p_2}^{tMin} \tag{3.7}$$

By the definition of paths p_1 and p_2, inequalities (3.2), (3.3), (3.4), and (3.5) are satisfied immediately. Inequalities 3.2 and 3.3 are satisfied because path p_2 visits the same nodes with path p_1 except node k. Furthermore, inequality (3.5) holds since the triangular inequality is assumed for the travel times for all arcs belonging to set A. Therefore, the rollback pruning strategy inspects only the inequalities (3.1), (3.6), and (3.7). The condition (3.6) holds if path p_1 requires less time to recharge

the battery fully compared to path p_2 or alternatively the remaining energy level of path p_1 is more than that of path p_2. Lastly, inequality (3.7) states that path p_1 spends less time compared to path p_2 for recharging and traveling. In Figure 3.1. , a representation of paths p_1 and p_2 are provided.

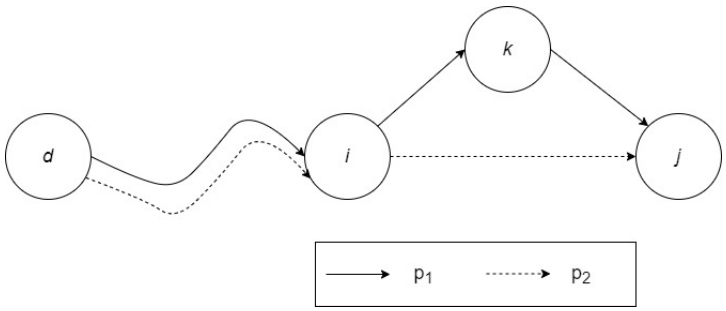

Figure 3.1. Rollback pruning strategy example

The depth-first search is vulnerable to the poor decisions made at the early stages of the exploration, since they may lead to the examination of unfavorable regions of the solution space. The rollback pruning strategy is specifically designed for the depth-first search. The method helps the algorithm catch these poor decisions sooner. Moreover, the efficiency of the pruning strategies impacts the performance of the labeling procedure strongly. Mainly because whenever a path is eliminated with a pruning strategy, not only a single solution but also an entire region of the solution space is discarded.

The Pulse algorithm does not have any dominance rules. Note that dominance is often the most time-consuming part of the labeling algorithms. It requires storing many labels related to the existing partial paths. On the other hand, working together with the dept-first search mechanism, the rollback procedure can be completed without the need of keeping all the labels. The algorithm runs on the same path until it is pruned, or it reaches the depot. Therefore, at each step, only the resources of the currently explored path are stored. More specifically, if the algorithm is sending a pulse through path p_1 in Figure 3.1. which travels through nodes i, k, and j consecutively, only the labels of path p_1 at nodes j and i are stored. The resources at node i are used when path p_1 enters the rollback function. This additional information at node i can be used to extend path p_2 to node j.

3.1.1. Bounding Procedure

Several relaxation methods are commonly used in the VRP literature to fathom sub-optimal paths. Bounding methods often enhance the relaxation of the consumption of some resources and focus on producing lower bounds based on a single resource. The bounding method suggested by Baldacci et al. (2011) generates a bound matrix $B(r,t)$ which is obtained by finding conditional lower bounds for every node i and for the discrete values of time consumption. The procedure relaxes the capacity constraint and assumes that the vehicles have an unlimited freight capacity.

Algorithm 3.3: Bounding Procedure

```
1  τ ← l_0;
2  while τ ≥ e_0;
3   do
4      τ ← τ − δ;
5      for  i ∈ N;
6       do
7          p = {};
8          c̄_p = 0;
9          q_p = 0;
10         T_i^{tMin} = τ;
11         T_i^{tMax} = τ;
12         T_i^{rtMax} = 0;
13         pulse(p, c̄_p, T_i^{tMin}, T_i^{tMax}, T_i^{rtMax});
14         if φ = {} then
15            B(i, τ) = ∞;
16         else
17            B(i, τ) = c̄_p;
```

The method represented by Algorithm 3.3 is employed as follows. Given that l_0 is the upper bound of the time window at the depot and δ is a nonnegative time step, the ESPPRC is solved for every node $i \in N$ given the exploitation starts at $t = l_0 - \delta$. At the first iteration, since only δ units of time is available, the resulting feasible regions for each node are noticeably small, and there are just a few feasible solutions. Thus, these problems can be rapidly solved to optimality with the *pulse* procedure (see Algorithm 3.2). The lower bound for node i is the minimum reduced cost of any path at node i which can use maximum δ units of time. Hence, at this iteration $\delta \geq T_p^{tMin} \geq T_p^{tMin}$ for each path p. After this step, the available time consumption is increased to 2δ, and bounds are found for every node given the starting time of $l_0 - 2\delta$. Although the size of the feasible region is increased, the lower bounds found in the first iteration can be used to fathom unpromising paths

at this step. This process is repeated for each increase of δ units until the starting time reaches e_0, which is the lower bound of the time window at the depot. The set ϕ in line 14 of Algorithm 3.3 represents the set of routes generated by the function *pulse*.

3.1.2. Improvements in Branch-and-Price Procedure

The Pulse algorithm is employed within the BP procedure which is enhanced with the acceleration methods explained in Section 2.3.4. These methods include IMP, ICP, AN, and terminating the route generation prematurely. The IMP generates upper bounds by solving the master problem at the root node of the BP tree. The ICP stores the complete routes that have not been sent to the RLPMP. There are two parameters related to the ICP, n^{col} and n^{iter}, which denote the limit on the number of routes stored in the pool and the maximum number of iterations that each column can be stored, respectively. The AN method has been proven to be efficient for the BP procedure provided in Chapter 2. It generates the set of non-dominated stations, F_{ij}, for each arc (i, j), $i \in V_0, j \in V_{n+1}$. Then, at node i, the path p is extended to node j directly or through a station $s \in F_{ij}$ (see Section 2.3.4.4).

The subproblem is terminated prematurely when the number of routes at the depot with negative reduced costs reaches a predetermined threshold value, n^{rt}. The algorithm described by Lozano et al. (2015) focuses on finding the route with the minimum reduced cost. Therefore, they do not stop solving an iteration of the pricing subproblem until they find all routes with negative reduced costs. However, solving the pricing problem is usually the bottleneck operation for BP procedures (Feillet et al., 2004; Kallehauge et al., 2005; Baldacci et al., 2012). Generating all feasible routes significantly increases the computation time that each pricing subproblem iteration requires, whereas sending only one route to RLPMP may increase the number of CG iterations. Considering these difficulties, we determined the threshold value for limiting the number of routes with negative reduced costs which are sent to RLPMP by implementing preliminary tests on a subset of instances.

3.1.3. Branching

We employ the branching procedure by applying the following strategies in the respective order. (Kohl et al., 1999; Kallehauge et al., 2005; Desaulniers et al., 2016): branching (i) on the total number of vehicles, (ii) on the total number of recharges, and (iii) on the total flow on an arc. When the CG ends with a fractional solution, we employ the first attainable rule using the provided order of branching rules. We refer the interested reader to the Section 2.3.2 for the details of the branching procedure.

Branching on the flow value of a specific station is excluded since our preliminary tests show that applying this rule not only causes more branches to be needed but also increases the amount of data stored.

3.2. Computational Study

In this section, we provide the results of the computational study that we performed using the BP procedure with the Pulse algorithm. For the experiments, we used JAVA (using IntelliJ Idea, 2020) language and linear programming models were solved by IBM ILOG CPLEX 12.9. The computer was on Windows 10 and has an Intel Core i7 processor running 3.2 GHz with 32 GB RAM allocation. All experiments were completed on instances with 25, 50, and 100 customers generated by Desaulniers et al. (2016) and the time limit was set to one hour. Our BP method terminated whenever all branches were solved or fathomed, or the gap between the best upper and lower bounds referred to as the relative integrality gap was less than a threshold value. The latter setting was applied to be consistent with Desaulniers et al. (2016). All average computation times were computed only for the instances solved to optimality within the time limit.

In what follows, we provide two sets of computational experiments. First, we present the results of the preliminary tests conducted to determine the parameters related to the bound on the number of routes with negative reduced cost and the ICP. For these experiments, the AN method was not included in the Pulse algorithm. The parameter tuning starts with determining the bound on the number of routes. We expect that the impact of this parameter on the algorithm performance is greater since it directly affects the computation time required for an iteration of the pricing subprob-

lem. Second, we provide a computational study performed by using all instances in 100-series (Desaulniers et al., 2016). Both single-partial (SP) and multiple-partial (MP) problem variants were solved and the impact of the AN method was evaluated.

3.2.1. Parameter Tuning

In this section, we first determined the threshold value on the number of complete routes with negative reduced costs at the depot, which is used to prematurely terminate the pricing subproblem. The experiments were conducted for the MP problem by using subsets of instances with 50 and 100 customers which included two instances from each of the C, RC, and R types. When determining n^{rt}, we set the values of ICP parameters to $n^{col} = 50$ and $n^{iter} = 2$ as suggested in Section 2.4.

In Table 3.1. , we present the best upper bound (Bound), the computation time in seconds (t), and the relative integrality gap (Δ) for each selected instance where n^{rt} is set to 2000, 1000, and 500. Additionally, the table provides the number of instances solved (#solved), average computation times, and average relative integrality gaps. The relative integrality gap is the gap between the best upper (z^*) and the best lower (z^{lb}) bounds and is calculated as $100\frac{z^*-z^{lb}}{z^*}$.

Table 3.1. Results of the preliminary tests for n^{rt} using 100-customer instances

Instance	2000			1000			500		
	Bound	t (s)	Δ (%)	Bound	t (s)	Δ (%)	Bound	t (s)	Δ (%)
C101	104376	**349.7**	0.00	104376	461.1	0.00	104376	381.5	0.00
C105	102889	**399.6**	4.65	102950	667.3	4.71	102993	588.5	4.75
RC101	165702	3600.9	5.38	163757	74.6	4.25	163866	**50.8**	4.32
RC102	152344	3601.4	5.09	152146	**263.6**	4.97	152819	3601.2	5.39
R101	157500	**46.9**	0.58	158018	57.7	0.91	157575	50.7	0.63
R102	142606	**205.0**	0.45	142449	322.8	0.34	142645	439.6	0.47
All		1367.2	2.69		**307.8**	**2.53**		852.1	2.59
#solved		4			**6**			5	

Considering instances with 100 customers, setting the threshold value to 2000 allows the algorithm to obtain shorter run times for the instances belonging to C and R types. However, considering the averages, limiting the number of complete routes having negative reduced cost to 1000 is the most time-efficient option for the selected subset. This value helps the algorithm solve all six instances within the time limit. Thus, n^{rt} is set to 1000 for the rest of the computational experiments using the

instances with 100 customers.

Table 3.2. Results of the preliminary tests for n^{rt} using 50-customer instances

Instance	2000			1000			500		
	Bound	t (s)	Δ (%)	Bound	t (s)	Δ (%)	Bound	t (s)	Δ (%)
C102	77719	**798.7**	0	77719	914.6	0.00	77719	1302.0	0.00
C103	64581	**356.0**	4.42	63940	886.5	3.36	63839	3355.4	3.21
RC106	88950	3600.0	5.5	87024	**12.3**	3.40	86815	14.1	**3.17**
RC108	73980	3600.0	5.58	75190	3600.0	7.10	75190	3600.0	7.10
R105	83475	10.3	1.87	83261	**8.4**	**1.61**	83482	8.9	1.87
R106	79216	**27.9**	**2.31**	79922	35.0	3.17	79812	158.5	3.04
All		1398.8	3.28		**909.5**	3.11		1406.5	**3.07**
# solved		5			**6**			5	

In Table 3.2. , we provide the results of n^{rt} tuning tests for the subset of instances with 50 customers. Note that, only by setting n^{rt} to 1000 all instances could be solved within the time limit. Additionally, the algorithm runs at least 35% faster by setting n^{rt} to 1000. Thus, n^{rt} is fixed to 1000 also for the instances with up to 50 customers.

Next, we conducted experiments to assess the most effective values of the ICP parameters. First, for n^{col} the values 10, 25, 50, 100, and 1000 were tested while n^{iter} was set to 2 since this is the best value reported in Section 2.4. The results are demonstrated in Table 3.3. .

Table 3.3. Results of the preliminary tests for n^{col}

n^{col}	100-Cust			50-Cust		
	#solved	t (s)	Δ (%)	#solved	t (s)	Δ (%)
1000	**6**	372.0	**2.40**	4	1561.3	4.03
100	**6**	**307.8**	2.53	5	1427.2	3.57
50	3	1971.6	2.86	5	909.5	3.11
25	5	837.6	2.49	**6**	**281.9**	**2.87**
10				5	840.5	3.57

Table 3.3. shows the average computation times and the average relative integrality gaps obtained by setting n^{col} to 1000, 100, 50, 25, and 10. For the instances with 100 customers, the value 10 is omitted based on the results obtained by using the values 25 and 50. The results show that the best value for the n^{col} differs as the data size varies. Indeed, for instances with 100 customers, we obtain the best results by setting n^{col} to 100 (this setting has an average time approximately 21% shorter than the second-best setting, $n^{col} = 1000$), while n^{col} is set to 25 for the instances with up to 50 customers.

In Table 3.4. , we present the results of the preliminary tests conducted to determine n^{iter}. The values one, two, and five were evaluated by using the best values for n^{rt} and n^{col} determined by the earlier experiments. The results favor the value two, so this value is fixed for the remaining experiments.

Table 3.4. Results of the preliminary Tests for n^{iter}

n^{iter}	100-Cust			50-Cust		
	#solved	t (s)	Δ (%)	#solved	t (s)	Δ (%)
1	5	761.8	2.52	5	947.4	3.32
2	**6**	**307.9**	**2.53**	**6**	**281.9**	**2.87**
5	5	834.7	2.73	5	844.5	3.38

We report the average integrality gap values for all tables, to evaluate the solution quality with different parameters and make sure that it does not deteriorate. The results show that there is a slight difference in the gap values. Although our average results include the non-optimal solutions as well, we observe that the average relative integrality gap values are small. This proves that the algorithm produces good upper bounds for the selected instances and they may be solved to optimality with new acceleration methods.

The parameter tests provided one more MP solution for RC103 with 100 customers, using the parameters $n^{col} = 50$ and $n^{iter} = 2$. The literature has not reported any solution for this instance, and we could not solve it with other parameters either. The instance was solved within 2183 seconds and the obtained solution had a 4.84% relative integrality gap.

The results of the instances with 25 customers fluctuated less compared to other sets when the parameters were changed. Hence, for these instances instead of conducting further tests to find values of the the parameters, the values determined for the 50-customer instances were used.

Table 3.5. summarizes the values selected by the parameter tuning experiments for all instances.

Table 3.5. Determined parameters

Set	n^{rt}	n^{col}	n^{iter}
25-Cust	1000	25	2
50-Cust	1000	25	2
100-Cust	1000	100	2

,

3.2.2. Experimental Results

In this section, we provide the results of the numerical study performed on the Pulse algorithm using instances with 25, 50, and 100 customers and the parameter values determined above. The Pulse algorithm was implemented using approaches with and without the AN method, and both methodologies were tested on all instances (Desaulniers et al., 2016) for two problem variants (SP and MP).

Table 3.6. provides a summary of the numerical results obtained from the evaluation of the Pulse algorithm. Pulse_0 and Pulse_{AN} denote the Pulse algorithm without and with the AN method, respectively. Columns #solved, t(s) and Δ (%) indicate the number of instances solved to optimality, the average computation time in seconds, and the average relative integrality gap, respectively.

Table 3.6. Summary of the results

			Pulse_0			Pulse_{AN}		
Problem Variant	n	Type	#solved	t (s)	Δ (%)	#solved	t (s)	Δ (%)
MP	25	All	29/29	142.5	1.03	29/29	57.9	0.88
		C-set	9/9	438.0	0.74	9/9	172.0	0.98
		RC-set	8/8	3.9	1.22	8/8	4.2	0.79
		R-set	12/12	13.4	1.12	12/12	8.1	0.86
	50	All	25/29	130.5	1.59	24/29	347.5	1.72
		C-set	8/9	204.4	0.67	7/9	258.6	0.89
		RC-set	7/8	59.1	2.13	6/8	103.9	2.05
		R-set	11/12	116.9	1.97	11/12	523.0	2.17
	100	All	14/29	491.2	2.96	10/29	1160.3	2.60
		C-set	3/9	783.4	3.12	3/9	1957.5	3.04
		RC-set	3/8	132.8	4.21	2/8	498.5	3.64
		R-set	8/12	515.9	2.44	5/12	946.7	1.92
SP	25	All	29/29	44.4	0.99	29/29	30.5	0.76
		C-set	9/9	133.2	0.48	9/9	89.1	0.48
		RC-set	8/8	3.2	1.16	8/8	3.1	0.75
		R-set	12/12	5.4	1.27	12/12	4.9	0.98
	50	All	24/29	264.6	2.31	25/29	197.2	2.24
		C-set	8/9	280.6	1.59	7/9	227.3	1.18
		RC-set	6/8	40.4	2.30	6/8	96.3	1.97
		R-set	10/12	386.4	2.88	12/12	230.2	2.97
	100	All	11/29	642.8	2.31	10/29	1024.4	1.55
		C-set	3/9	1134.1	0.97	4/9	1582.4	0.88
		RC-set	3/8	270.5	4.02	3/8	443.7	3.55
		R-set	5/12	571.3	2.42	3/12	861.0	1.12

All 25-customer instances are solved optimally and the average computation times

obtained by using Pulse_{AN} are 31% and 59% shorter in MP and SP cases, respectively compared to those obtained by using Pulse_0. Considering MP case over the instances with 50 customers, the AN approach solves one less instance compared to Pulse_0 and runs 62% slower. On the other hand, in the SP case over the 50-customer instances, Pulse_{AN} solves one more instance to optimality compared to Pulse_0 and runs 25% faster on the average. Considering 100-customer instances, the Pulse algorithm with AN procedure runs 1.6 and 2.4 times slower in MP and SP cases, respectively, and shows an inferior performance by solving four fewer instances in MP and one less in SP compared to Pulse_0. In sum, although the AN approach enhanced the performances of HCG and TPA in all data sets in Chapter 2, we observe that it is not as efficient when combined with the Pulse algorithm for larger instances. It can be predicted that this low performance is due to the combination of the AN method and dept-first search approach used in the route creation procedure.

Table 3.7. Comparison of the performances of Pulse algorithm and monodirectional algorithms from the literature

			Pulse$_0$			TPA			Desauniers et. al. 2016		
Problem Variant	n	Type	#solved	t (s)	Δ (%)	#solved	t (s)	Δ (%)	#solved	t (s)	Δ (%)
MP	25	All	29/29	142.5	1.03	29/29	14.7	0.94	29/29	24.4	0.80
		C	9/9	438.0	0.74	9/9	18.8	0.46	9/9	1.3	0.43
		RC	8/8	3.9	1.22	8/8	6.1	1.12	8/8	2.0	0.89
		R	12/12	13.4	1.12	12/12	17.4	1.19	12/12	56.7	1.02
	50	All	26/29	130.5	1.59	26/29	342.3	1.73	26/29	380.5	1.17
		C	8/9	204.4	0.67	8/9	162.8	0.87	8/9	46.4	0.28
		RC	7/8	59.1	2.13	7/8	234.5	2.62	7/8	209.1	2.06
		R	11/12	116.9	1.97	11/12	541.3	1.78	10/12	732.6	1.26
	100	All	14/29	491.2	2.96	8/29	950.3	2.12	10/29	419.6	1.67
		C	3/9	783.4	3.12	3/9	843.2	3.01	3/9	372.2	2.18
		RC	3/8	132.8	4.21	2/8	1319.5	2.55	3/8	361.5	2.53
		R	8/12	515.9	2.44	3/12	811.2	1.09	4/12	498.6	0.64
SP	25	All	29/29	44.4	0.99	29/29	7.1	0.79	29/29	2.6	0.66
		C	9/9	133.2	0.48	9/9	9.0	0.52	9/9	1.1	0.47
		RC	8/8	3.2	1.16	8/8	3.2	0.87	8/8	1.4	0.59
		R	12/12	5.4	1.27	12/12	8.2	0.94	12/12	4.5	0.86
	50	All	24/29	264.6	2.31	26/29	211.0	1.99	25/29	130.2	1.37
		C	8/9	280.6	1.59	8/9	60.3	1.42	8/9	17.8	0.88
		RC	6/8	40.4	2.30	6/8	150.0	1.73	7/8	50.3	1.83
		R	10/12	386.4	2.88	12/12	342.0	2.50	10/12	276.1	1.45
	100	All	11/29	642.8	2.31	10/29	857.8	1.66	9/29	247.6	0.91
		C	3/9	1134.1	0.97	5/9	768.3	1.26	6/9	149.1	0.95
		RC	3/8	270.5	4.02	1/8	1877.5	4.85	1/8	1291.6	1.77
		R	5/12	571.3	2.42	4/12	714.7	1.38	2/12	21.1	0.38

Since the overall performance of Pulse_0 is better than that of Pulse_{AN}, especially in the instances with 100 customers, we compare it with the results of the monodirectional algorithms from the literature namely the TPA provided in Chapter 2 and the BPC algorithm proposed by Desaulniers et al. (2016). Our observations on the results are summarized in Table 3.7. are as follows:

- All instances involving 25 customers are solved optimally by all algorithms. However, the Pulse algorithm is significantly slower than the others.
- Considering the instances with 50 customers, the Pulse algorithm solves at least one less instance compared to others for both MP and SP cases. Nevertheless, in MP problem, it achieves the shortest average computation time of all.
- In the MP case with instances including 100 customers, the Pulse algorithm solves six and four more instances compared to TPA and the BPC, respectively. Moreover, it runs on the average 52% faster compared to TPA and only 7% slower compared to the BPC algorithm.
- In the SP problem and instances with 100 customers, the Pulse algorithm solves one and two more instances compared to TPA and the BPC algorithm, respectively. It runs on average 25% faster than TPA, but around two times slower than the BPC algorithm.
- The results also demonstrate the efficiency of the Pulse algorithm on the randomly located data sets such that the Pulse algorithm solves at least as many R-type instances as the others for each experiment.

One reason for the higher performance of the Pulse algorithm on R-type of instances can be related to the setting in which *ng-route* relaxation is not implemented. Considering *ng-route* based BP algorithms, visiting a customer multiple times can lead the algorithm to return to the promising neighborhoods. Thus, the *ng-route* algorithm provides excellent results for the instances that include geographically clustered customers (C-type). However, the method can be less efficient for R-type instances since the chance of returning to unpromising neighborhoods is often greater for R-type compared to C-type.

In comparison with the other exact algorithms, the Pulse algorithm is particularly effective for the MP case with 100-customer instances.

We also evaluated the performance of the Pulse algorithm by using one branching rule only on the MP problem instances. The efficiency of the hierarchical branching procedure proposed here can be evaluated by benchmarking the results obtained by

Pulse_0 with the results of the experiments including only one branching rule. For the instances with up to 50 customers, the performances of the two approaches are similar. However, for the 100-customer instances Pulse_0 containing three branching rules runs on the average 30% faster compared to the version including one branching rule.

Table 3.8. New and improved solutions

Problem Variant	n	Instance	BKS	Bound	Δ (%)	t (s)
MP	100	RC101	163949	163757	4.25	63.1
SP	100	RC101	n/a	173326	4.45	68.2
SP	100	RC106	140885	140022	4.85	1877.5

Table 3.8. presents the results of the instances for which either the BKS is improved or a solution is introduced to the literature for the first time. The fourth column reports BKS values reported in Chapter 2. The best solution value obtained by Pulse_0 is provided in the fifth column. The last column reports the computation time in seconds. The full set of results is provided in Appendix B.

3.3. Conclusion

In this chapter, we modified the Pulse algorithm to solve the EVRPTW. The algorithm was improved by using the following acceleration methods: (i) terminating the labeling algorithm when the number of efficient routes at the depot exceeds a threshold value, (ii) applying an Intermediate Column Pool (ICP) to store routes that are sent to RLPMP yet, (iii) implementing the Integral Master Problem (IMP) at the root node of the Branch-and-Price (BP) to improve the upper bound and (iv) employing the Augmented Node (AN) method to eliminate visits to dominated stations. Moreover, a branching procedure was implemented by including three rules: branching on the number of routes, on the number of recharges, and on the arc-flow values.

The computational experiments include two main parts: the preliminary tests for determining the most suitable values of the parameters and the complete experiments which analyze the performance of the algorithm for all instances and two problem variants. We reported the results of the preliminary tests conducted to determine the ICP parameters and the bound on the complete routes for stopping the pricing subproblem prematurely. After determining the values of these parameters,

the algorithm with and without the AN approach was evaluated for all and both problem cases. The results show that the AN method implemented in the Pulse algorithm is effective only for small-sized data sets.

The results of the Pulse algorithm without using the AN approach, Pulse_0, were compared to those obtained by Desaulniers et al. (2016) and the results favor Pulse_0 in both MP and SP problem over the instances with 100 customers. In addition, Pulse_0 introduced three new solutions to the literature. The numerical experiments certify that even though the Pulse procedure has similarities with labeling algorithms, the specific pruning mechanisms like the bounding method and rollback pruning provide smarter exploitation of the problem network.

The methods proposed in this chapter can be further improved by implementing the network reduction techniques to solve larger problem instances. Additionally, future research may also focus on solving extensions of the EVRPTW such as the versions considering multiple recharge technologies and uncertainties related to service and recharging times. The Pulse algorithm can be improved using parallel computing techniques and valid inequalities.

4. A BRANCH-AND-PRICE ALGORITHM FOR THE ELECTRIC VEHICLE ROUTING PROBLEM WITH FLEXIBLE DELIVERY

In this chapter, we propose an effective Branch-and-Price (BP) algorithm to solve the Electric Vehicle Routing Problem with Flexible Delivery (EVRP-FD) introduced by Sadati et al. (2022). In the EVRP-FD, the fleet comprises of identical electric vehicles (EVs) and customers can be associated with several delivery locations for each of which a time window is provided. Each day the EVs are dispatched from a central depot with full batteries to serve each customer exactly once at one of their locations within the provided time window. The EVs can be recharged at charging stations that are assumed to be uncapacitated but scarce. The aim of the EVRP-FD is to minimize the total distance traveled while satisfying capacity and time window constraints.

To the best of our knowledge, Sadati et al. (2022) is the only study related to the EVRP-FD in the literature. The authors formulate a mathematical model and develop a hybrid variable neighborhood search method together with a tabu search mechanism to solve it. Additionally, the study introduces new instances for the EVRP-FD and provides a computational study evaluating the performance of the algorithm on these instances and investigating the effect of several parameters such as fixed cost, freight capacity, charger type, and battery capacity on the solution characteristics.

Little research has been conducted which addresses the VRP with flexible delivery options. The first study related to the concept of flexible options for delivery is presented by de Jong et al. (1996). The article addresses the VRP with Multiple Time Windows (VRPMTW) in which customers can provide alternating time windows for the delivery. Then, Doerner et al. (2008) develop a branch-and-bound method and a heuristic algorithm to solve the VRPMTW and evaluate them by using a data set including instances with up to 15 customers. Later, the problem is also addressed by Favaretto et al. (2007), Belhaiza et al. (2014), and Beheshti et al. (2015) where metaheuristic algorithms are proposed to solve it.

The concept of flexible delivery locations is introduced as the VRP with Roaming Delivery Locations (VRPRDL) by Reyes et al. (2017). In the VRPRDL, customers' orders are delivered to one of their locations within the related time window. The problem is also known as "*trunk delivery*" in which the demands of the customers can be delivered to the trunk of their car which may be located in different places at different periods of the day. The study also introduces the VRP with Home and Roaming Delivery Locations (VRPHDL) serving the customers either at their home locations anytime during the day or at one of the other locations within the corresponding time window. The study suggests a two-phase heuristic method to solve both problems. The objective is to minimize the total traveling cost which is linearly related to the total distance traveled. Additionally, a new data set is generated and the results of an extensive computational study on these instances are included in the study.

Ozbaygin et al. (2017) presents a BP algorithm to solve the VRPRDL and the VRPHDL. The authors apply the state-space augmentation method (Boland et al., 2006) along with the label setting algorithm suggested by Feillet et al. (2004) to solve the pricing subproblem of CG. The BP procedure is advanced with various acceleration methods including premature termination of the labeling algorithm and storage of non-dominant routes that were not sent to the linear relaxation of the constrained main problem. Also, to deal with the tailing effect (the decrease in the rate of improving the bounds towards the end of CG), the authors recommend using Lagrange dual bounds and premature termination of CG. The performance of the algorithm is evaluated by using both the instances provided by Reyes et al. (2017) and newly generated 20 instances consisting of two variations of instances with 40 customers.

A generalized version of the VRPRDL, the VRP with Delivery Options (VRPDO) is addressed by Tilk et al. (2021). In this problem, the customers have multiple delivery options such as collecting their delivery from lockers and shops, using a reception box and controlled access systems, and requesting a trunk delivery. The authors develop a Branch-Price-and-Cut (BPC) algorithm to solve the problem. In addition, a bidirectional *ng-route* algorithm is proposed to solve the pricing subproblem of the BPC. Two sets of valid inequalities are proposed to reduce the integrality gap faster: *k-path* inequalities (Kohl et al., 1999) and subset-row inequalities (Jepsen et al., 2008). The study provides instances including up to 100 options and a computational study evaluating the performance of the BPC procedure with the new instances. Moreover, the results obtained for the VRPRDL and the VRPHDL are also reported and the performance of the BPC algorithm is compared to that of the solution procedure provided by Ozbaygin et al. (2017).

We develop a BP algorithm employed with a Column Generation (CG) procedure to solve the EVRP-FD. A bidirectional Pulse algorithm (Lozano et al., 2015) is proposed to solve the pricing subproblem of the EVRP-FD. The algorithm needs to preserve only a few labels during a CG iteration because of the depth-first search-based exploitation of the digraph. The suboptimal paths can be eliminated by one of the following strategies at the early stages of the exploitation: (i) the bounding method which creates a bound matrix using lower bounds on the reduced costs found for each node and discrete values of resource consumption, (ii) rollback pruning method reevaluating whether the last node included to the path should be removed, and (iii) infeasibility pruning method eliminating infeasible paths.

Our BP algorithm is promoted by employing several acceleration techniques: the bidirectional search, IMP, ICP, and heuristic pricing (see Section 2.3.4). The efficiency of the bidirectional search is widely benefited for solving shortest path problems (Pohl, 1971; Luby and Ragde, 1989; Righini and Salani, 2006). The second technique, IMP, generates an upper bound for by solving the MILP formulation of the master problem at the root node of the BPC tree. Moreover, the ICP stores the routes with non-negative reduced costs which can be used in later iterations when the dual variables change (Taş et al., 2014). We start each iteration of the CG by employing the heuristic column generator (HCG) explained in Section 2.3.4.

The contributions of this chapter are summarized as follows.

- We provide a MILP formulation for the EVRP-FD and propose a BP algorithm improved with the state-of-the-art methods to solve it.
- We present a modified version of the bidirectional Pulse algorithm explained in Chapter 3 to solve the pricing subproblem of the CG algorithm. Additionally, we improve the performance further by applying HCG before employing the exact CG procedure.
- We present new benchmark instances with up to 120 customers, 470 locations, and 13 stations. In addition, we provide an extensive computational study evaluating the performance of the proposed procedure by using these instances.

The remainder of this chapter is organized as follows. Section 4.1 describes the problem in detail. Section 4.2 explains the BP procedure. Section 4.3 provides the details of the experimental study and discusses the numerical results. Finally, concluding remarks are provided in Section 4.4.

4.1. Problem Description

In the EVRP-FD, the customers whose demands, alternative delivery locations, and related time windows are provided are served by using a homogeneous fleet of EVs. EVs are dispatched from the depot at the beginning of each day with fully charged batteries and return to the depot at the end of their tour. The fleet must respect the time windows of the locations and the scheduling horizon represented by the time window of the depot. The stations are identical in terms of charging speed and cost and can be visited multiple times by different EVs. For the sake of simplicity, we assume that the battery charging time is linearly proportional to the amount of energy transferred. In addition, EVs are allowed to visit at most one station along their route. This is a rational assumption since recharging the battery of EVs more than once during the planning horizon may not be practical in urban logistics operations.

The problem is defined on a complete digraph in which $V = \{1, ..., n\}$ denotes the set of all locations and A is the set of all arcs. The depot is represented by node 0 for the departures and by $n + 1$ for the arrivals. The set of customers is denoted by V_c. The locations related to customer c are denoted by L_c. The demand of customer c, q_c, should be satisfied within one of the specified non-overlapping time windows $[e_i, l_i]$ where $i \in L_c$. Note that q_c is equal to q_i for all $i \in L_c$. If an EV arrives at location i before the earliest possible service time e_i, it waits until e_i. On the other hand, no arrivals after the latest possible service time, l_i, are accepted for location i.

The distance along arc (i, j) is denoted by d_{ij} where the triangular inequality is satisfied. In case a vehicle visits a station s while traveling from node i to node j we introduce d_{ijs} that represents the total distance traversed as $d_{ijs} = d_{is} + d_{sj} - d_{ij}$. Additionally, t_{ij} denotes the time spent for traveling directly from node i to node j where $t_{ijs} = t_{is} + t_{sj} - t_{ij}$.

The decision variables τ_i^k and y_i^k keep track of the start time of service and the battery state of charge (SoC) of vehicle k at its arrival at node i, respectively. In case vehicle k travels from node i to node j through station s, the decision variables y_{ijs}^k and Y_{ijs}^k represent the battery SoC of that vehicle at its arrival at station s and departure from station s, respectively. The binary decision variable x_{ij}^k takes the value 1 if vehicle k traverses arc (i, j) directly or through a station, and 0 otherwise. Furthermore, z_{ijs}^k is a decision variable which takes the value 1 if vehicle k traverses arc (i, j) through station s, and 0 otherwise. The mathematical notation

is summarized in Table 4.1. .

Table 4.1. Mathematical notation

Sets	
V	Set of locations, $V = \{1, .., n\}$
V_c	Set of customers,
L_c	Set of locations of customer, $c \in V_c$
V_0	Set of locations and the departure depot, $V_0 = V \cup \{0\}$
V_{n+1}	Set of locations and the arrival depot, $V_{n+1} = V \cup \{n+1\}$
$V_{0,n+1}$	Set of locations, the departure depot and the arrival depot, $V_{0,n+1} = V_0 \cup \{n+1\}$
F	Set of stations
K	Set of EVs
Parameters	
d_{ij}	Distance along arc $(i, j) \in A$
t_{ij}	Travel time along arc $(i, j) \in A$
q_i	Demand of location i, $i \in V$
$[e_i, l_i]$	Service time window at node i, $i \in V_{0,n+1}$
C	Vehicle freight capacity
Q	Vehicle battery capacity
g	Battery charging rate
h	Energy consumption rate
t_{ijs}	Detour time spent for visiting station s while traveling from node i to node j, $t_{ijs} = t_{is} + t_{sj} - t_{ij}$, $i \in V_0$, $j \in V_{n+1}$, $s \in F$
d_{ijs}	Detour distance traversed by visiting station s while traveling from node i to node j, $d_{ijs} = d_{is} + d_{sj} - d_{ij}$, $i \in V_0$, $j \in V_{n+1}$, $s \in F$
Decision Variables	
τ_i^k	Service start time of vehicle k at customer i, $i \in V$, $k \in K$
y_i^k	Battery State of Charge (SoC) of vehicle k at its arrival at node i, $i \in V_{0,n+1}$, $k \in K$
Y_{ijs}^k	Battery SoC of vehicle k at its departure from station s if it travels from node i to node j through station s, $i \in V_0$, $j \in V_{n+1}$, $s \in F$, $k \in K$
y_{ijs}^k	Battery SoC of vehicle k at its arrival at station s if it travels from node i to node j through station s, $i \in V_0$, $j \in V_{n+1}$, $s \in F$, $k \in K$
x_{ij}^k	1, if vehicle k traverses arc (i, j), directly or through a station, 0 otherwise, $i \in V_0$, $j \in V_{n+1}$, $k \in K$
z_{ijs}^k	1, if vehicle k travels from node i to node j through station s, 0 otherwise, $i \in V_0$, $j \in V_{n+1}$, $s \in F$, $k \in K$

Below, we present a modified version of the EVRPTW formulation presented in Section 2.2.

$$\min \quad \sum_{i \in V_0} \sum_{j \in V_{n+1}, j \neq i} \sum_{k \in K} \left(d_{ij} x_{ij}^k + \sum_{s \in F} d_{ijs} z_{ijs}^k \right) \tag{4.1}$$

subject to

$$\sum_{k\in K}\sum_{i\in L_c}\sum_{j\in V_{n+1},j\neq i} x_{ij}^k = 1, \qquad c \in V_c, \quad (4.2)$$

$$\sum_{i\in V_0,i\neq j} x_{ij}^k - \sum_{i\in V_{n+1},i\neq j} x_{ji}^k = 0, \qquad k \in K, j \in V_{0,n+1}, j \neq i, \quad (4.3)$$

$$\sum_{s\in F} z_{ijs}^k \leq x_{ij}^k, \qquad k \in K, i \in V_0, j \in V_{n+1}, \quad (4.4)$$

$$\tau_i^k + t_{ij}x_{ij}^k + \sum_{s\in F}\Big(t_{ijs}z_{ijs}^k + g(Y_{ijs}^k - y_{ijs}^k)\Big) - l_0(1-x_{ij}^k) \leq \tau_j^k, \quad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (4.5)$$

$$e_j \leq \tau_j^k \leq l_j, \qquad k \in K, j \in V_{0,n+1}, \quad (4.6)$$

$$\sum_{i\in V} q_i \sum_{j\in V_{n+1},j\neq i} x_{ij}^k \leq C, \qquad k \in K, \quad (4.7)$$

$$0 \leq y_j^k \leq y_i^k - hd_{ij} + (Q+hd_{ij})\left(1 - x_{ij}^k + \sum_{s\in F} z_{ijs}^k\right), \quad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (4.8)$$

$$y_j^k \leq \sum_{s\in F}(Y_{ijs}^k - hd_{sj}z_{ijs}^k) + Q\Big(1 - \sum_{s\in F} z_{ijs}^k\Big) + Q(1-x_{ij}^k), \quad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (4.9)$$

$$y_{ijs}^k \leq y_i^k - hd_{is}z_{ijs}^k + Q(1-x_{ij}^k), \qquad k \in K, s \in F, i \in V_0, j \in V_{n+1}, i \neq j, \quad (4.10)$$

$$0 \leq y_{ijs}^k \leq Y_{ijs}^k \leq Qz_{ijs}^k, \qquad k \in K, s \in F, i \in V_0, j \in V_{n+1}, i \neq j, \quad (4.11)$$

$$\sum_{s\in F}\sum_{i\in V_0}\sum_{j\in V_{n+1},j\neq i} z_{ijs}^k \leq 1 \qquad k \in K, \quad (4.12)$$

$$x_{ij}^k \in \{0,1\}, \qquad k \in K, i \in V_0, j \in V_{n+1}, i \neq j, \quad (4.13)$$

$$z_{ijs}^k \in \{0,1\}, \qquad k \in K, i \in V_0, j \in V_{n+1}, s \in F, i \neq j. \quad (4.14)$$

The objective (4.1) is to minimize the total distance traveled. Constraints (4.2) guarantee that customers are visited exactly once in one of their locations. Flow balance is guaranteed by constraints (4.3). Constraints (4.4) ensure that at most one station is visited between each pair of locations. Constraints (4.5) enforce the time feasibility of arcs emanating from the locations and the depot. Constraints (4.6) ensure that each node is served within its time window. Additionally, constraints (4.7) assure that the vehicle capacity is not exceeded. Battery SoC consistency throughout the route is satisfied by constraints (4.8) - (4.10). In particular, constraints (4.8) guarantee the battery feasibility at locations and the depot. Constraints (4.9) guarantee the battery feasibility at node j if a station is visited between nodes i and j and determine the amount of energy transferred to the battery. Battery feasibility on the arrival at the stations is provided by constraints (4.10). If a station is not visited, constraints (4.11) guarantee that the related charging variables take

the value of 0. Constraints (4.12) limit the number of recharges on each route by one. Constraints (4.13) and (4.14) define the binary decision variables.

4.2. Solution Methodology

The BP algorithm is initialized by applying the preprocessing rules and the greedy construction algorithm reported in Section 2.3 to eliminate infeasible arcs from the problem network and to find an initial feasible solution, respectively.

We develop a BP method in which a CG algorithm is solved at each branch. The master problem of the CG is the set partitioning formulation given by (4.15) - (4.17). In this formulation, the set of all feasible routes starting from the departure depot (node 0) and ending at the arrival depot (node $n+1$) is represented by Ω. c_p denotes the cost of route p and a_{pc} is equal to 1 if route p visits a location of customer c and 0 otherwise. Moreover, θ_p is a binary decision variable that takes the value of 1 if route p is part of the solution and 0 otherwise.

$$\min \quad \sum_{p \in \Omega} c_p \theta_p \tag{4.15}$$

$$\text{subject to} \quad \sum_{p \in \Omega} a_{pc} \theta_p = 1, \qquad c \in V_c, \tag{4.16}$$

$$\theta_p \in \{0, 1\}, \qquad p \in \Omega. \tag{4.17}$$

The objective function (4.15) minimizes the total cost. Constraints (4.16) ensure that exactly one location of each customer is visited. Lastly, constraints (4.17) define the binary decision variables. Only a subset of routes, Ω_R, is evaluated at each CG iteration (see Section 2.3 for the detailed description of CG).

The MILP formulation (4.1)-(4.14) can be separated for each vehicle k by using Danzig-Wolfe decomposition. This problem denotes the elementary shortest path problem with resource constraints. The CG starts by solving the Restricted Linear Programming Master Problem (RLPMP) where only the routes generated by the initial feasible solution are included in Ω_R. At each CG iteration, the pricing subproblem aims at producing routes with negative reduced costs that can improve the current solution and utilizes the values of the dual variables obtained from the

RLPMP to calculate the reduced costs of paths. The reduced cost $\bar{c}_p$ associated with route p is calculated as $\bar{c}_p = c_p - \sum_{c \in V_c} a_{pc}\pi_c$ where π_c is the value of the dual variable associated with the set partitioning constraint of customer c. The CG ends when the labeling algorithm cannot produce any more routes with negative reduced costs.

4.2.1. Solving the Pricing Subproblem

We propose the Pulse algorithm (Lozano et al., 2015) to solve the pricing subproblem of the BP method. The Pulse algorithm explores the network and generates partial paths until they reach the depot or are eliminated by a pruning strategy. It consists of two general steps: the first stage involves a bounding procedure that finds lower bounds on the reduced cost given an amount of resource consumption for each node. In the second step, the routes with negative reduced costs are recursively generated and the solution space is implicitly enumerated.

After applying the bound method, the algorithm begins exploring the location-based digraph at the depot node, travels throughout the outgoing arcs of each visited node using depth-first search structure, and stores the resulting partial path p, the current node i, the reduced cost $\bar{c}_p$, and the cumulative consumption values for each resource. The first resource W_i^{load} is the freight load of path p. Note that, the demand of customer c can be considered as the demand of location i, where $i \in L_c$. The second resource is time and measured by using a combination of three time related resources, T_i^{tMin}, T_i^{tMax}, and T_i^{rtMax} which are related to the arrival time to node i under different recharging rules and recharging time, as in the generic labeling algorithm provided in Chapter 2.

The Pulse algorithm utilizes three pruning strategies to discard suboptimal partial paths at the early stages of the search. The first strategy is infeasibility pruning which eliminates a path if it is infeasible. The infeasibility occurs if one of the following situations happens: more than one location of a customer is visited, the arrival time considering battery feasibility is later than the upper bound of the time window of the location i, $T_i^{tMin} > l_i$, the arrival time considering battery feasibility is later than the arrival time considering time window feasibility $T_i^{tMin} > T_i^{tMax}$, the cumulative load exceeds load capacity of the vehicles C, $W_i^{load} > C$, and the maximum required recharging time is greater than the recharging time for an empty battery, $T_i^{rtMax} > gQ$.

The second pruning strategy is a bound check procedure that eliminates paths if the summation of their reduced costs and the lower bound obtained with the bounding procedure is greater than a continuously updated global bound on the reduced cost of a complete route, $\bar{c}_p*$. The bounding method finds conditional lower bounds on the reduced cost of paths for every node and for all discrete values of time consumption.

The last pruning strategy is called rollback pruning which checks the last choice made in the following fashion. Consider a path p_1 from the depot to node i which travels to node k and arrives at node j. The rollback pruning strategy reevaluates the decision of visiting node k before node j in the following way. An alternative path p_2 visits the same nodes in the same order with p_1 until node i and then it travels to node j.

We refer the interested reader to Section 3.1 for the details of the bounding procedure, rollback pruning strategy, and the exploitation procedure.

4.2.2. Branching

When the CG ends with a fractional solution, two new branches are created from the currently solved tree node. Among the arcs which have a fractional flow value, we select the arc having a flow value close to 0.5. The total flow for each arc is calculated as $f_{ij} = \sum_{k \in K} x_{ij}^k$ where x_{ij}^k is the flow of arc (i, j) of route operated by vehicle k and computed by using the RLPMP solution (Kallehauge et al., 2005). More details are provided in Section 2.3.2.

The branch-and-bound tree is enumerated using the depth-first search strategy since several studies in the literature state that it is more effective compared to the breadth-first search for solving the VRPTW variants with a BP algorithm (Taş et al., 2014; Desaulniers et al., 2016).

4.2.3. Acceleration Methods

The Pulse algorithm is employed with the BP procedure which is enhanced with several acceleration methods including the IMP, ICP, and terminating the route generation prematurely. We use IMP to generate an upper bound by solving the integer formulation of the master problem when the root node of the BP tree is

completely solved or the time limit has been reached before the root node of the BP tree is not completed. In the second case, the IMP provides a feasible solution. We report the results of this approach separately in the computational study to provide a reference for future research.

ICP stores the complete routes that have not been sent to the RLPMP. There are two parameters related to the ICP: n^{col} and n^{iter} denote the limit on the number of routes stored in the pool and the maximum number of iterations that each column can be stored, respectively. At each iteration of CG, the reduced cost of each route in ICP is recomputed by using the last dual variables obtained by the RLPMP. If the algorithm finds routes with negative reduced costs, they are sent to RLPMP which is then run again.

The subproblem is terminated prematurely when the number of routes at depot with negative reduced costs reaches a predetermined threshold value, n^{rt}. We determine the value of this parameter to limit the number of routes with negative reduced costs sent to the RLPMP by implementing preliminary tests on a subset of instances.

In the rest of this section, the details of the bidirectional search mechanism and the heuristic labeling algorithm are provided.

4.2.3.1. Bidirectional Search

The details of the forward search are provided in Section 4.2.1. Additionally, we implement a backward search that starts from the arrival depot and finishes at the departure depot. Each path starts at time l_0 and the time is decreased as nodes are visited.

Let $L_p = (W_i^{load}, W_i^{tMin}, W_i^{tMax}, W_i^{rtMax}, w_p, \mathbf{B}_p^V)$ be the label of the backward path p. The definitions of W_i^{load}, w_p and $\mathbf{B}_p^V$ are similar to those given for T_i^{load}, a_p and $\mathbf{V}_p^V$, respectively (see Section 4.2.1). Moreover, W_i^{tMin}, W_i^{tMax} and W_i^{rtMax} are used to manage the arrival time and the battery capacity as before. The related formulation is the same as the formulation provided in Section 2.3.4.3 for backward labeling.

Forward paths and backward paths are extended until a predefined time $M \in (e_0, l_0)$. The search stops and the label of the path p is saved when T_i^{tMin} and W_i^{tMin}, are less than or equal to M for forward and backward labeling, respectively. A forward label and a backward label at a node are combined if the conditions (4.18)-(4.21)

hold.

$$T_j^{load} + W_j^{load} - q_j \leq C, \tag{4.18}$$

$$V_p^i + B_p^i \leq 1 \ \forall i \in V_{0,n+1} \backslash \{j\}, \tag{4.19}$$

$$T_i^{tMin} + Z_i \leq W_i^{tMin}, \tag{4.20}$$

$$Z_i \leq (T_i^{tMax} - T_i^{tMin}) + (W_i^{tMin} - W_i^{tMax}) \tag{4.21}$$

where $Z_i = \max(0, T_i^{rtMax} - W_i^{rtMax} - gQ)$ denotes the time required at a station for recharging the minimum amount of energy to ensure that the battery SoC is never negative on the route (Desaulniers et al., 2016). Condition (4.18) states that the total load of a vehicle cannot exceed its freight capacity. Condition (4.19) ensures that each location is not visited more than once. Condition (4.20) guarantees that the time window of node i is respected. Condition (4.21) states that the minimum required recharging time cannot exceed the available route time. Note that this condition is not checked if node i represents a recharging station.

4.2.3.2. Heuristic Pricing

Heuristic column generator (HCG) is an *ng-route* (Baldacci et al., 2012) based method (see Section 2.3.4 for the details related to HCG). We propose an enhanced CG procedure starting with the HCG to effectively determine a good upper bound to the RLPMP. First, the HCG produces routes that feed the RLPMP until no more columns with negative reduced costs can be found. Then, the bidirectional Pulse algorithm is employed as long as the algorithm generates at least one route with a negative reduced cost.

4.3. Computational Study

In this section, we provide the results of the extensive computational study evaluating the performance of our BP procedure. We evaluated the performance of the BP algorithm on all instances and two problem variations. These variations were obtained according to the VRPRDL and the VRPHDL addressed by Reyes et al.

(2017) and Ozbaygin et al. (2017). We solved extensions of these problems by utilizing a fleet of EVs. Although our main study regards single (partial) recharge (SP) per route, we also performed experiments with multiple-partial (MP) case which allows recharging more than once en route. Nevertheless, for the MP problem, we assume that only one station can be visited between each customer pair, since traveling from one station to another for recharging is not practical in urban logistics. For MP problem, constraints 4.4 is added to the pricing subproblem and constraints 4.12 are removed from it.

We used JAVA (using IntelliJ Idea, 2020) language to implement the BP algorithm. The linear programming models were solved by using IBM ILOG CPLEX 12.9 (IBM, 2020). The computer was on Windows 10 Pro and had an Intel Core i9 processor running 3.6 GHz with 64 GB RAM allocation.

In the following, first a detailed explanation of data generation is provided. Then, we present the results of the parameter tuning tests applied to determine the values of the parameters associated with the bound on terminating the subproblem prematurely and with the ICP. Next, we provide detailed solutions for the EVPR-FD and EVRP-FDH and report the results of the experiments performed with the MP relaxation. Finally, we present the feasible solutions obtained for the instances that cannot be solved by using the BP procedure. The computational time was limited to two hours for the instances with up to 60 customers, and 12 hours for the instances with 120 customers. Additionally, the triangular inequality was obtained by solving all-pairs shortest path problem before the BP procedure begins as suggested by Ozbaygin et al. (2017).

4.3.1. Data Generation

For the VRPRDL, the literature provides two benchmark data sets reported by Reyes et al. (2017) (Set 1) and Ozbaygin et al. (2017) (Set 2). In both sets, customers start and finish the day at their home locations. In addition, a customer can be linked to a maximum of six locations. In Set 1, the locations were placed randomly in a circle around the customer's home location with a predefined radius. The set consists of 40 instances with the following sizes: 5 instances with 15 customers, 5 instances with 20 customers, 10 instances with 30 customers, 10 instances with 60 customers, and 10 instances with 120 customers. Moreover, the instances with 60 customers have up to 237 locations whereas instances with 120 customers include up to 470 locations.

Set 2 was generated to investigate the effect of the distances of service locations to the depot and consists of two variations of instances with 40 customers (each variant has 10 instances). The first variation of these instances was obtained by using the same instance generation mechanism as Reyes et al. (2017). In the second variation, the instance generation procedure was not altered for the radius of the circular area on which the locations are placed randomly. However, the center of the area was moved to the middle of the line connecting the customer's home location to the warehouse. This increases the distance between the home location and the other locations for each customer. Thus, the time windows were tightened to make sure customers can visit all of their locations and return to home locations during the planning period. More specifically, the width of the time windows in Set 1 are 23% wider on average compared to that of Set 2.

We generated the EVRP-FD instances by benefiting from the properties of both Set 1 and Set 2 instances described above.

We assume that the fleet considered for the experiments consists of Mercedes EQV (Mercedes-Benz, 2021). This EV has a driving range of 300 km and a battery energy capacity of 90 kWh that can be effectively used. Therefore, the discharging rate is computed as $90/300 = 0.3$ kWh/km. We consider chargers with 90kW power.

The time horizon is set to 12 hours in the instances provided by Ozbaygin et al. (2017). However, the data set includes some customers whose goods cannot be delivered to any of their locations within 12 hours by using a Mercedes EQV, since these locations are placed more than 150 km away from the depot and EVs need to recharge at least once to complete a round trip starting from the depot and visiting these locations next. In such circumstances, the total route time including the recharging time exceeds 12 hours. We eliminate this infeasibility by setting the time horizon to 13 hours in all instances and shifting the time windows of the location that has the latest time window closing.

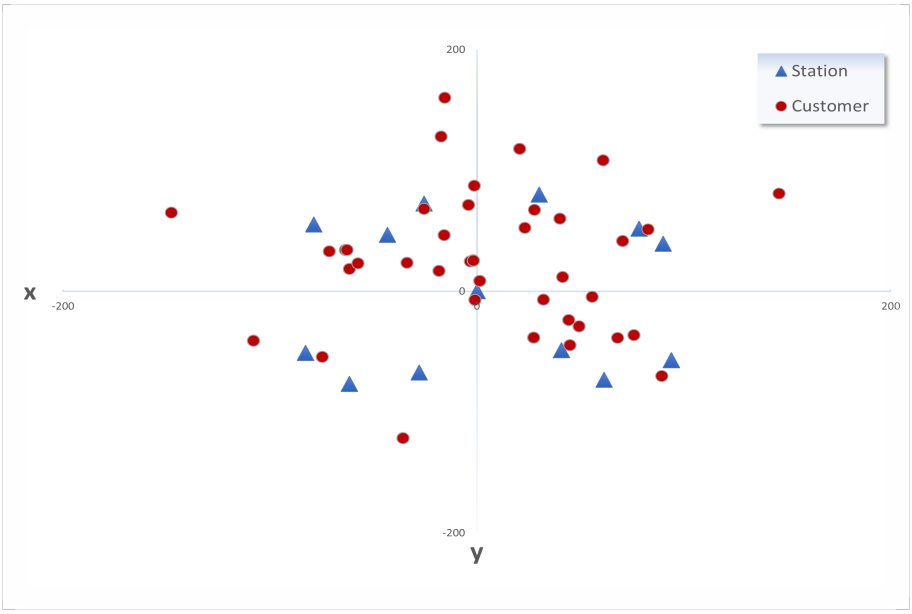

Figure 4.1. Representation of stations on the Cartesian coordinate system

We included recharging stations into the data in the following fashion. Uniform random numbers are used to generate x and y coordinates of the stations for which we consider that a round trip starting from the depot and visiting a station is feasible with a full battery. We placed three stations at each region in the Cartesian coordinate system and there is a charging station at the depot. The locations of these 13 stations are fixed for all data sets. A representation of the locations of the stations and the customers of a randomly selected instance is provided in Figure 4.1. .

4.3.2. Parameter Tuning

We conducted preliminary tests to determine the limit on the number routes sent to the RLPMP, n^{rt}, and to obtain the values of the ICP parameters n^{col} and n^{iter}. We only considered the EVPR-FD and a subset of six instances where we randomly selected four instances from Set 1 (two instances with up to 30 customers and two instances with 60 customers) and two instances from Set 2 (with 40 customers).

Table 4.2. displays the results of the tests conducted for the parameter n^{rt}. Columns Bound, t (s) and Δ (%) indicate the best upper bound, the average computational time in seconds, and the relative integrality gap, respectively. The relative integrality gap is the gap between the best upper (z^*) and the best lower (z^{lb}) bounds and is calculated as $100\frac{z^*-z^{lb}}{z^{lb}}$. Note that, the bound and the relative integrality gap

are not displayed for each test separately in the tables provided in this section since they remain the same for different values of the parameters. Moreover, when an instance could not be solved optimally within the time limit, the run time of two hours was considered when computing the average run time.

n^{rt} was determined as five for the bidirectional labeling algorithm applied to solve the EVRPTW (see Section 2.4). Hence, we started the evaluation by assessing $n^{rt} = 5$ and considered the values $n^{rt} = \{5, 10, 20, 50\}$. While tuning n^{rt}, the ICP parameters n^{col} and n^{iter} was fixed to 50 and 2 as suggested in Section 3.2.

Table 4.2. Results of the preliminary tests conducted to determine n^{rt}

Instance	Bound	Δ (%)	t (s)			
			n^{rt}=5	n^{rt}=10	n^{rt}=20	n^{rt}=50
E10	1395	0.00	1.2	1.3	**0.6**	1.5
E19	1992	4.35	8.6	7.9	**7.6**	9.1
E21	3060	0.00	315.0	305.9	**160.3**	711.7
E26	2882	3.72	224.1	181.5	**169.4**	248.7
E41.v1	2706	3.65	63.0	99.1	**23.4**	37.1
E43.v2	1848	0.27	619.1	482.4	749.0	**445.4**
All		2.00	205.2	229.5	**185.0**	280.6

According to Table 4.2. , setting $n^{rt} = 20$ provides the smallest average computational time, that is 11%, 24%, and 51% shorter compared to that obtained when n^{rt} is set to 5, 10, and 50, respectively. Thus, we set n^{rt} to 20 for the following computational study.

Table 4.3. Results of the preliminary tests conducted to determine n^{col}

Instance	Bound	Δ (%)	t (s)		
			n^{col}=25	n^{col}=50	n^{col}=100
E10	1395	0.00	1.3	**0.6**	1.2
E19	1992	4.35	7.9	**7.6**	8.2
E21	3060	0.00	291.8	**160.3**	323.9
E26	2882	3.72	227.2	**169.4**	180.4
E41.v1	2706	3.65	80.2	**23.4**	99.6
E43.v2	1848	0.27	7200.0	**749.0**	1206.3
All		2.00	1301.4	**185.0**	303.3

Table 4.3. provides the results of the tests conducted for the parameter n^{col} by using the same subset of instances. $n^{col} = 50$ was determined by the parameter tuning test in Section 3.2. Therefore, we considered the values $n^{col} = \{25, 50, 100\}$ and the smallest average computational time was obtained when we set n^{col} to 50.

We further observe that the average run time of the algorithm is 86% and 39% shorter when n^{col} equals 50 compared to that obtained when the parameter is set to 25 and 100, respectively.

Table 4.4. Results of the preliminary tests conducted to determine n^{iter}

Instance	Bound	Δ (%)	t (s)		
			n^{iter}=1	n^{iter}=2	n^{iter}=3
E10	1395	0.00	1.2	0.6	1.0
E19	1992	4.35	**7.2**	7.6	12.9
E21	3060	0.00	170.5	**160.3**	634.8
E27	2882	3.72	223.0	169.4	**218.6**
E41.v1	2706	3.65	35.0	23.4	**32.3**
E43.v2	1848	0.27	7200.0	749.0	604.8
All		2.00	1272.8	**185.0**	252.4

Lastly, we present the results of tests conducted to determine the value of the parameter n^{iter} in Table 4.4. . n^{iter} was set to 2 according to the parameter tuning tests presented in Section 3.2. For the EVRP-FD, we considered $n^{iter} = \{1, 2, 3\}$ and the results favor when n^{iter} is equal to 2. Note that, the algorithm solves one less instance when $n^{iter} = 1$ leading to 85% longer computational time compared to the case when n^{iter} is set to 2.

We observe that n^{iter} have been set to 2 by all parameter tests in this dissertation. Hence, we can conclude that this value is robust for n^{iter} and it is not altered when the problem or the solution method changes. A similar conclusion can be made for n^{col}. The value of 50 provides sufficiently good solutions for the instances including at least 100 nodes in all EVRP variants and solution methods considered in this dissertation.

4.3.3. Experimental Results for the EVRP-FD

In the following four tables (including two tables in the next section), we provide the results for two separate experiments. The first experiment, denoted as Test-1, was conducted by using the approach of Desaulniers et al. (2016) in which the algorithm is terminated if one of the following conditions occurs: (i) all branches are solved or fathomed, (ii) the time limit is reached, or (iii) the relative integrality gap is less than the threshold value. The second experiment (Test-2) was carried out to find a better bound for the instances solved by Test-1 if the solution obtained by Test-1

had a non-zero relative integrality gap. Therefore, for Test-2, we did not provide a threshold value for the relative integrality gap and ran the algorithm until (i) or (ii) held.

The related results of the EVPR-FD instances are displayed in Tables 4.5. and 4.6. , for Set 1 and Set 2, respectively. Additional to the columns described in the earlier section, #n and #b represent the number of customers and the total number of branches used in the solution, respectively.

As Table 4.5. displays that all instances with up to 60 customers were solved within the time limit by Test-1 and the average run time is 264.2 seconds. 17 of these solutions are proven to be optimal since their relative integrality gaps are zero. In addition, three instances with 120 customers were solved within 22513 seconds on average.

Furthermore, we present optimal solutions for six instances with up to 60 customers obtained by Test-2 and improved upper bounds for four instances that were already solved by Test-1. Moreover, the Test-1 solution of E11 was proven to be optimal by Test-2 (the upper bound in Test-1 is the same as the optimal solution obtained by Test-2).

Table 4.5. Results of the EVRP-FD for Set 1

		Test-1				Test-2			
Instance	**#n**	**Bound**	**#b**	***t* (s)**	**Δ (%)**	**Bound**	**#b**	***t* (s)**	**Δ (%)**
E1	15	880	0	1.9	0.00				
E2		1140	0	0.7	0.00				
E3		998	0	0.7	0.00				
E4		1037	0	0.7	0.00				
E5		1838	0	0.3	0.00				
E6	20	1303	0	1.6	1.76	1302	2456	506.3	0.00
E7		1178	0	4.2	3.92	1164	4647	7200.0	2.68
E8		1366	0	2.3	0.00				
E9		1282	0	4.0	3.72	1266	988	229.2	0.00
E10		1395	0	0.6	0.00				
E11	30	1928	0	6.6	0.63	1928	1354	1489.3	0.00
E12		1802	0	8.2	0.00				
E13		1678	0	43.0	0.00				
E14		1191	0	9.7	0.00				
E15		1611	0	6.1	3.85	1606	7316	7200.0	3.17
E16		1931	0	11.0	0.00				
E17		1985	0	5.89	0.88	1982	850	1102.1	0.00
E18		1834	0	5.0	0.00				
E19		1992	0	7.6	4.34	1978	2242	7200.0	3.53
E20		1642	0	9.1	1.08	n/a			
E21	60	3060	0	160.3	0.00				
E22		2743	0	1411.5	0.27	2737	4	4087.4	0.00
E23		3655	0	2436.3	0.00				
E24		3335	0	109.4	0.94	3333	369	7200.1	0.88
E25		2911	0	805.8	0.00				
E26		4339	0	90.0	0.00				
E27		2882	0	169.4	3.72	n/a			
E28		3447	0	2436.4	0.00				
E29		3468	0	99.3	3.74	n/a			
E30		3893	0	77.5	0.07	3891	2	202.3	0.00
E32	120	4875	0	9791.3	0.64	n/a			
E33		4556	0	29923.1	1.30	n/a			
E35		4881	0	27823.9	1.84	n/a			

In Table 4.6. , we report the results of the experiments for the instances in Set 2. Considering Test-1, all instances were solved within the time limit and the average computational time is 489.3 seconds. On the other hand, Test-2 solved four more instances to optimality within an average of 1169.4 seconds. Furthermore, for three more instances, it improved the upper bounds generated by Test-1 even though the algorithm was not terminated within the run time limit.

Table 4.6. Results of the EVRP-FD for Set 2

	Test-1				Test-2			
Instance	**Bound**	**#b**	***t* (s)**	**Δ (%)**	**Bound**	**#b**	***t* (s)**	**Δ (%)**
E41.v1	2706	0	23.4	3.65		n/a		
E42.v1	2611	0	32.8	0.29	2606	16	292.0	0.00
E43.v1	2220	0	108.0	2.73		n/a		
E44.v1	2114	0	100.1	2.62	2100	677	7200.0	1.94
E45.v1	2519	0	90.6	0.04	2518	4	161.0	0.00
E46.v1	2485	0	174.4	0.00				
E47.v1	2640	0	31.6	3.60		n/a		
E48.v1	3091	0	20.2	4.26		n/a		
E49.v1	3188	0	10.9	0.00				
E50.v1	2239	0	530.0	1.28	2223	11	7200.0	0.56
E51.v2	2141	0	56.0	4.97	2111	1034	7200.0	3.50
E52.v2	1908	0	76.8	1.43	1895	230	3294.4	0.00
E53.v2	1848	0	749.0	0.27	1844	2	930.1	0.00
E54.v2	1675	0	595.1	4.12		n/a		
E55.v2	2250	0	5062.3	0.00				
E56.v2	2360	0	216.6	0.16		n/a		
E57.v2	1898	0	321.7	0.00				
E58.v2	2285	0	542.0	0.00				
E59.v2	2391	0	162.0	1.49		n/a		
E60.v2	2194	0	882.3	0.40		n/a		

To conclude, the BP algorithm solved all EVRP-FD instances (in Sets 1 and 2) with up to 60 customers within 354.2 seconds by applying Test-1. This test provided optimal solutions for 55% of those instances and the average relative integrality gap is 2.15% over the instances for which a feasible solution could be obtained. Combining both tests, we provide optimal solutions for 80% of the instances with up to 60 customers.

4.3.4. Experimental Results for the EVRP-FDH

Tables 4.7. and 4.8. provide the results obtained for the EVRP-FDH over the instances in Set 1 and Set 2, respectively.

According to the results presented in Table 4.7. , 19 instances with up to 30 customers were solved within the time limit with Test-1 and the average run time is 454.6 seconds. 10 of those instances were proven to be optimal since their relative integrality gaps were zero. In addition, the average relative integrality gap is 2%

for the instances for which a feasible solution could be obtained. Moreover, two instances with 60 customers were solved within 5874 seconds on average.

Test-2 includes six additional optimal solutions and their average computational time is 1600.2 seconds. Additionally, the upper bound found by Test-1 for E18 was improved by Test-2.

Table 4.7. Results of the EVRP-FDH for Set 1

		Test-1				**Test-2**			
Instance	**#n**	**Bound**	**#b**	***t* (s)**	**Δ (%)**	**Bound**	**#b**	***t* (s)**	**Δ (%)**
E1	15	777	0	6.0	2.44	773	144	121.9	0.00
E2		1066	0	2.5	0.00				
E3		998	0	3.1	0.00				
E4		827	0	2.9	1.45	825	422	172.1	0.00
E5		1748	0	1.2	4.02	1714	7232	642.2	0.00
E6	20	1077	0	4.6	2.04	1077	8224	6604.2	0.00
E7		995	0	41.8	0.75	994	114	896.4	0.00
E9		998	0	29.1	0.00				
E10		1110	0	38.5	0.00				
E11	30	1522	0	683.6	1.34	1522	194	7200.0	0.98
E12		1602	0	932.0	0.00				
E13		1563	0	308.3	0.00				
E14		1076	0	406.8	0.84	1068	2	1164.3	0.00
E15		1289	0	1718.9	0.00				
E16		1293	0	2909.4	0.00				
E17		1460	0	147.1	0.00				
E18		1585	0	166.7	4.30	1571	886	7200.0	3.08
E19		1448	0	216.2	2.08		n/a		
E20		1575	0	260.6	0.00				
E21	60	2534	0	4742.4	0.72		n/a		
E27		2373	0	7005.4	1.19		n/a		

In Table 4.8. , we report the solutions of the EVRP-FDH over the instances in Set 2. The BP algorithm applied with the Pulse algorithm solved 12 instances with Test-1. The average computational time of these solutions is 1949.7 seconds. The relative integrality gaps are zero for five instances (optimal solutions) whereas the average of that is less than 1% for the remaining instances (feasible solutions). Further, we introduce an additional optimal solution by using Test-2.

Overall, the BP algorithm solved 80% of the EVRP-FDH instances with up to 30 customers to optimality. 63% of those solutions were obtained by Test-1.

Table 4.8. Results of the EVRP-FDH for Set 2

	Test-1				Test-2			
Instance	**Bound**	**#b**	***t* (s)**	**Δ (%)**	**Bound**	**#b**	***t* (s)**	**Δ (%)**
E41.v1	2454	0	112.3	0.00				
E42.v1	2369	0	239.5	0.00				
E45.v1	2489	0	542.3	0.00				
E46.v1	2456	0	1978.0	0.00				
E47.v1	2250	0	1230.2	3.62		n/a		
E48.v1	2810	0	451.8	0.21		n/a		
E49.v1	3146	0	362.0	0.00				
E51.v2	1949	0	3567.6	0.41		n/a		
E56.v2	2210	0	4513.3	1.04		n/a		
E57.v2	1758	0	6391.5	0.29		n/a		
E58.v2	2235	0	780.2	0.72	2228	2	7200.0	0.41
E59.v2	2328	0	3227.9	0.09		n/a		

The EVRP-FDH has wider time windows in general. It is known that solving the instances with tight time windows is easier than solving those with wide time windows (Dell'Amico et al., 2006; Bettinelli et al., 2014). Hence, the algorithm performed better for the EVRP-FD than the EVRP-FDH, considering the average computational times and the number of instances solved.

4.3.5. Summary of Branch-and-Price Experiments

We present a summary of the results of the computational experiments in Table 4.9. , where the columns #n, #solved, #optimal, #ImpBound, and Δ (%) represents the number of customers, the number of instances solved, the number of optimal solutions, the number solutions improved by Test-2, and the average relative integrality gap, respectively.

Considering both problem types, we evaluated our algorithm on 100 instances with up to 60 customers and solved 85 of them. In total 88% of the EVRP-FD instances were solved including all instances in Set-1 and Set-2 with up to 60 customers and three instances with 120 customers. Moreover, Test-1 and Test-2 provide optimal solutions for 77% of the Set-1 instances with up to 60 customers.

Considering the EVRP-FDH, we provide solutions for 85% of the instances with up to 30 customers and 129 locations, 60% of the Set 2 instances, and two instances

with 60 customers. Including the six optimal solutions found by Test-2, in total 89% of the instances with up to 30 customers are proven to be optimal.

Table 4.9. Summary of results obtained by the branch-and-price algorithm

			Test-1			Test-2		
Type	Set	#n	#solved	#optimal	Δ (%)	#ImpBound	#optimal	Δ (%)
EVRP-FD	Set 1	≤ 30	20/20	12	1.01	6	4	0.49
		60	10/10	5	0.89	3	2	0.83
		120	3/10	0	2.73	0	0	n/a
	Set 2	40 (V1)	10/10	2	1.85	4	2	0.62
		40 (V2)	10/10	3	1.28	3	2	0.91
EVRP-FDH	Set 1	≤ 30	19/20	10	0.97	6	6	0.18
		60	2/10	0	0.96	0	0	n/a
	Set 2	40 (V1)	7/10	5	0.55	0	0	n/a
		40 (V2)	5/10	0	0.49	1	0	0.40

4.3.6. Results of the Multiple Recharge Experiments

In this section, we disregarded the restriction that each EV is allowed to be charged only once, to investigate the effect of having multiple recharges (MP) on the performance of the BP algorithm and the quality of the solutions. We provide the related solutions in Table 4.10. for the instances for which MP obtained a better solution. As in earlier sections, we performed two experiments denoted Test-1 and Test-2 representing the experiments including the threshold value for the relative integrality gap to terminate the algorithm prematurely and not including it, respectively.

We introduce 22 solutions in total for both EVRP-FD and the EVRP-FDH cases by using the MP relaxation. Considering the EVRP-FD, for two instances namely E44.v1 and E48.v1, MP relaxation provides a better bound than the SP case in both experiments. Moreover, Test-1 achieved better bounds than the SP case for six additional instances including three instances in Set-2 (E43.v1, E56.v2, and E59.v2) and three instances with 120 customers that could not be solved for the SP problem (E34, E38 and E40). For the instances with 120 customers, the average computational time is 21026 seconds and the relative integrality gap is 2.73% on average. Furthermore, Test-2 solved six EVRP-FD instances in total and improves the SP solutions for all of them.

Table 4.10. Results of the improved solutions with MP relaxation

			Test-1				Test-2			
Type	Set	#n	#solved	#b	t (s)	Δ (%)	#solved	#b	t (s)	Δ (%)
EVRP-FD	Set 1	E7	1178	0	7.0	3.97	1151	698	565.3	0.00
		E9	1297	635	330.9	4.97	1269	5069	2827.9	0.00
		E13	1676*	0	8.1	0.00				
		E20	1647	0	27.4	1.32	1639	1272	7200	0.83
		E23	3653*	0	751.3	0.00				
		E24	3330	0	159.4	0.74		n/a		
		E34	4924	0	22033.1	3.40		n/a		
		E38	4577	0	23358.5	0.81		n/a		
		E40	4520	0	17686.5	3.98		n/a		
	Set 2	E43.v1	2219	0	494.9	2.79				
		E44.v1	2110	0	128.3	2.58	2101	621	7200.0	2.11
		E48.v1	3088	0	43.9	4.43	3087	1476	7200.0	4.40
		E54.v2	1681	8	748.8	4.23	1656	342	7200.1	2.68
		E56.v2	2352	0	786.6	0.00				
		E59.v2	2386	0	109.9	1.26		n/a		
EVRP-FDH	Set 1	E21	2529	0	3374.7	0.52				
		E30	2461	0	5598.1	0.00				
	Set 2	E47.v1	2183*	0	3620.1	0.69		n/a		
		E48.v1	2804	0	6230.1	0.00				
		E51.v2	1941	0	4810.1	0.00				
		E59.v2	2327	0	806.7	0.00				

* Solution includes routes with multiple recharges

The MP problem provides an additional optimal solution for an EVRP-FDH instance, E30, that could not be solved in the SP case. In addition, the bounds of five instances are improved by the first MP experiment (Test-1) and three of those instances were solved to optimality.

Considering both experiments, we present optimal solutions for nine instances with the MP problem. The optimal solutions of three instances including the EVRP-FD instances E13, E23, and the EVRP-FDH instance E47.v1 contain routes with two recharges. In the remaining solutions, all EVs visit at most one station during the day. Removing the recharge restriction increases the search space for the algorithm, resulting in better bounds for 26% of all instances solved with SP experiments, and only three of those solved consist of routes with multiple station visits.

4.3.7. Results of the Heuristic Upper Bounding Experiments

In this section, we report the solutions obtained for the instances that cannot be solved by using the BP procedure to provide a feasible solution (bound) for the

following studies. We applied the IMP procedure to obtain an upper bound for an instance if algorithm stops due to time limit and could not find any upper or lower bounds. Shown by our preliminary experiments that applying IMP does not consume significantly long time since the master problem contains only the set-partitioning constraints. In Table 4.11. , we provide the corresponding results where columns Bound and Δ_i denote the upper bound found by the IMP and the percentage of the improvement from the initial solution, respectively.

Table 4.11. Upper bounds for the unsolved instances

Type	Set	Instance	Bound	Δ_i (%)
EVRP-FDH	Set 1	E22	2131	43.99
		E23	2757	51.55
		E24	2438	59.68
		E25	2244	54.53
		E26	2712	59.13
		E29	2629	53.67
		E30	2461	60.40
	Set 2	E43.v1	2071	56.14
		E44.v1	2001	52.09
		E50.v1	2269	52.58
		E53.v2	1811	44.57
		E54.v2	2794	6.24
		E55.v2	2207	47.07
		E60.v2	2059	43.56

We provide upper bounds for 14 EVRP-FDH instances in total. For an instance, the improvement from the initial solution is less than 10%. We obtained on average 51% improvement for the remaining instances.

4.4. Conclusion

In this study, we addressed a variant of the EVRPTW, the Electric Vehicle Routing Problem with Flexible Delivery (EVRP-FD). In the EVRP-FD, customers may request their orders to be delivered to one of the predetermined delivery locations within the corresponding time windows. Each day a homogeneous fleet of EVs is dispatched from a central depot to serve each customer exactly once in one of their locations within the provided time window. The EVs can be recharged at most once

a day in the recharging stations which are uncapacitated but scarce. This study is the first presenting a Branch-and-Price (BP) algorithm employed with a Column Generation (CG) procedure to solve the problem. We utilized the Pulse algorithm developed by (Lozano et al., 2015) to solve the pricing subproblem and enhanced it by implementing a bidirectional search. The bidirectional search starts both from the departure and arrival depots simultaneously which increases the efficiency of the BP procedure significantly.

The BP algorithm was improved by employing several acceleration techniques besides the bidirectional search including terminating the subproblem prematurely when the number of routes with negative reduced cost reaches a predetermined threshold, Integer Master Problem (IMP), Intermediate Column Pool (ICP), and the heuristic column generation procedure. Limiting the number of generated routes with negative reduced costs eases the algorithm since the pricing subproblem requires more computational effort than solving the RLPMP for our problem. Moreover, the IMP generating an upper bound for the problem at the root node of the BPC tree enables obtaining solutions for most of the instances at the root node of the BP tree. This method was also benefited when the BP algorithm could not find any bounds for an instance within the time limit. ICP holds complete routes with non-negative reduced costs which are reevaluated in the later CG iterations as the dual values are updated. Furthermore, heuristic pricing was employed with HCG to reduce the time spent on the exact labeling procedure.

In the computational study, we offered two data sets including 13 stations and up to 120 customers obtained by modifying the instances reported by Reyes et al. (2017) and Ozbaygin et al. (2017). We generated instances for EVRP-FD and EVRP-FDH variants by considering VRPRDL and VRPHDL cases developed by Reyes et al. (2017), respectively. We applied parameter tuning tests on the EVRP-FD instances to determine the values of the parameters associated with the bound on terminating the subproblem prematurely and the ICP. With the tuned parameters, our algorithm was evaluated by using both problem types and all instances. Two separate experiments were conducted with a difference in the terminating conditions of the algorithm. The branch-and-bound algorithm solved 88% of the EVRP-FD instances and 53% of the EVRP-FDH instances. Moreover, we improved the bounds of 22 instances by utilizing the MP relaxation in which the number of recharges per route is not limited. Lastly, we report upper bounds for 14 instances by solving an IMP at the root node for the instances that the BP could not obtain a solution.

When the MP and SP problem results are compared, the BP algorithm provided better bounds for 18% of the EVRP-FD instances with MP relaxation and four

additional solutions. Moreover, almost all of these solutions consist of routes with single recharging (only three instances contain routes with multiple recharging). Although those results show that allowing one recharge along a route is a realistic assumption, it might be slowing down the algorithm regarding the search through the promising neighborhoods. In conclusion, another approach for improving the performance of the BP can be running the Pulse algorithm without the charging limitation and not sending the routes with multiple recharges to the RLPMP.

For future research, we plan to employ valid inequalities and network reduction techniques. In addition, we consider the parallel application of the bidirectional Pulse algorithm. By means of the depth-first-search structure of the algorithm, the problem network can be exploited parallel using multiple threads. Moreover, a variant of the EVRP-FD which includes overlapping time windows for the customer locations can be addressed.

5. FINAL REMARKS

In this study, we addressed the EVRPTW and the EVRP-FD. We presented mathematical models for both problems, and adopted the state-of-the-art methods from the literature. More specifically, exact and heuristic algorithms based on the CG procedure that was embedded in BP or BPC schemes to effectively solve these problems. We developed novel procedures all based on the Pulse algorithm or the *ng-route* algorithm to solve the pricing subproblems of those methods. We provided extensive numerical studies evaluating the performances of the proposed procedures.

In Chapter 2, we presented an exact and a heuristic algorithm based on the BPC method to solve the EVRPTW. For both algorithms, the *ng-route* algorithm was benefited to solve the pricing subproblem. We generated an effective heuristic method by relaxing a label of the *ng-route* algorithm which significantly decreased the computation time of the algorithm. We proposed an efficient CG procedure for the exact method which consists of a heuristic labeling phase followed by the *ng-route* algorithm. The performances of the algorithms were improved by using the following methods: the bidirectional search during the route generation, the ICP that stores the routes with non-negative reduced costs to be used in the next iterations, the AN method that eliminates inefficient routes over the network, the bounding method that eliminates partial paths using lower bounds on reduced costs, and IMP method which helps to produce good upper bounds. Additionally, we limited the number of efficient routes since it speeds up solving the pricing subproblem which requires more computational effort than solving the RLPMP. Furthermore, we enhanced the BP by using a set of valid inequalities. We evaluated the algorithms on a well-known data set that includes instances with up to 100 customers, introduced 21 new solutions to the literature, and improved solutions of two instances that have already been provided in the literature.

In Chapter 3, we focused on the EVRPTW and developed a BP procedure for which the pricing subproblem was solved by an effective label setting method known as the Pulse algorithm. The algorithm was enhanced with the acceleration techniques

proposed in Chapter 2. In the computational study, we analyzed the performance of the algorithm with and without the AN approach for all instances for both single-partial (SP) and multiple-partial (MP) variants. The results showed that the AN method implemented in the Pulse algorithm is effective only for small-sized data sets. Moreover, the results of the Pulse algorithm without using the AN approach, Pulse_0, were compared to those obtained by Desaulniers et al. (2016) and the Two-Phased Algorithm (TPA) proposed in Chapter 2. The results favored Pulse_0 in both MP and SP variants over the instances with 100 customers since it solved several more instances compared to other algorithms. In addition, Pulse_0 introduced three new solutions to the literature. The numerical experiments certify that even though the Pulse procedure has similarities with labeling algorithms, the specific pruning mechanisms like the bounding method and rollback pruning provide smarter exploitation of the problem network.

In Chapters 2 and 3, we addressed the EVRPTW and proposed two different exact algorithms. The main difference between the solution mechanisms was the labeling procedure which provided varying performances for different data types. In the EVRPTW, there are three types of instances according to their geographic positions on the digraph: random (R-type), clustered (C-type), and randomly clustered (RC-type). We noticed that over the R-type of instances the performance of the Pulse algorithm was better than that of the algorithms based on the *ng-route* method. More specifically, the Pulse algorithm solved at least as many R-type instances as the other methods for each experiment. One reason for the better performance of the Pulse algorithm on R-type of instances can be presented as follows: in the *ng-route* based BP algorithms, visiting a customer multiple times can lead the algorithm to return to the promising neighborhoods and generate efficient routes faster. Thus, the *ng-route* algorithm provides excellent results for the instances that include geographically clustered customers (C-type). However, the method can be less effective for R-type instances since the chance of returning to unpromising neighborhoods is often greater for these instances compared to C-type. Since the Pulse algorithm does not rely on *ng-route* relaxation, it may be performing at least as good as the other for R-type of instances.

Comparing the effectiveness of our algorithms for SP and MP variants, we observed that limiting the number of recharges per route increased the performance of the *ng-route* based algorithms proposed in Chapter 2 for the instances with 100 customers. For this data set, both the TPA and the heuristic algorithm (monodirectional and bidirectional versions) solved three more instances on average for the SP case compared to the number of solutions they obtained for the MP problem. On the other hand, the BP with Pulse algorithm proposed in Chapter 3 solved three more in-

stances with 100 customers for the MP problem than for the SP problem. The reason for this can be related to the search mechanisms used by the algorithms search through the problem network. The *ng-route* based algorithms utilize a breadth-first search structure and they simultaneously enumerate many partial paths. Since the number of feasible routes increases in the MP variant, these algorithms need to keep more labels at the same time which deteriorates the performance of the algorithm directly. On the other hand, the Pulse algorithm uses a depth-first search structure and keeps only one path and the related label at a given moment. When a path visits a station for the second time the algorithm has to eliminate that path and start from the beginning, thus, the time spent on obtaining that path is wasted.

In Chapter 4, we addressed the EVRP-FD which provides the customers the flexibility to request their orders to be delivered to one of the predetermined locations within the relevant time window. This is a more complex problem compared to the EVRPTW since each customer can be associated with up to six locations. We proposed an exceptional bidirectional version of the Pulse algorithm embedded in a BP procedure to solve the EVRP-FD. The BP algorithm was improved by employing the state-of-the-art acceleration techniques including terminating the subproblem prematurely when the number of routes with negative reduced cost reaches a predetermined threshold, the IMP, the ICP, and the heuristic pricing with the Heuristic Column Generator (HCG). First the heuristic pricing procedure was applied and then the CG with the Pulse algorithm is employed since it was shown in Chapter 2 that HCG produces most of the efficient routes, and thus, significantly reduces the time spent for the exact labeling procedure. We offered a data set including 13 stations and up to 120 customers obtained by modifying benchmark instances. We evaluated the algorithm on these instances considering both SP and MP problem variants and provided solutions for 85% of the instances with up to 60 customers and up to 237 locations and 30% of the instances with 120 customers and up to 470 locations.

For the MP problem, the BP algorithm obtained better bounds for 18% of the EVRP-FD instances that are solved with the SP case and provided four additional solutions. Except for three instances, all these solutions consist of routes with single recharging. Since most of the routes in the solutions include only one visit to stations, allowing one recharge along a route is a realistic assumption. However, it might be slowing down the algorithm regarding the search through the promising neighborhoods. Thus, another approach for improving the performance of the BP can be running the Pulse algorithm without the charging limitation and not sending the routes with multiple recharges to the RLPMP.

In future studies, the BP and BPC-based methods (TPA, HCG, and Pulse algorithm) proposed in Chapters 2, 3 and 4 can be adapted to similar problems using fleets of EVs such as the EVRPTW with flexible time windows and variants with a capacity of charging stations. In this dissertation, we assumed that EVs cannot serve a customer if they arrive later than the upper bound of the time window of that customer. However, in reality, this assumption may be too strict to follow for carrier companies and a problem including soft time windows can be studied in which EVs may arrive after the upper bound of the time window but are punished with a penalty cost. Additionally, the assumption of the linear recharging function may not be applicable in real life and can be further investigated in the future for both EVRPTW and EVRP-FD. Moreover, it is possible to strengthen algorithms in various ways including generating the branch-and-bound tree using a best-first branching technique which aims to select the most promising node to branch and solve the pricing subproblem on a narrowed network first and apply the labeling algorithm on the full network next. In this dissertation, we assumed that the chargers are always available and the EVs can start recharging as soon as they arrive at a station. However, in practice, the chargers may be busy recharging other vehicles. Hence, the EVs may wait in queue for service. Future research within this context may investigate the stochastic nature of the problem by considering queueing times at stations. Furthermore, many companies prefer recharging their EVs at their depot or dedicated stations because of security concerns regarding the cargo onboard and driver idle times. Therefore, the problem can be extended to address the routing and charge scheduling decisions of the fleet simultaneously.

The BP algorithm proposed to solve the EVRP-FD may be improved by employing valid inequalities and implementing the bidirectional Pulse algorithm in a parallel structure which allows utilizing multiple threads for route generation. Using the depth-first-search structure of the algorithm, the problem network can be exploited parallel by using multiple threads. Furthermore, an interesting research direction is to address a variant of the EVRP-FD which includes overlapping time windows for the customer locations.

Appendix A. Detailed Results of the TPA and HCG

This appendix provides instance-based results of the computational analysis conducted to evaluate both monodirectional and bidirectional versions of the TPA and the HCG, and the multiple-partial recharge (MP) and the single-partial recharge (SP) problem variants over the complete dataset. The columns Bound, Δ_{BKS} (%), #b, t(s), and Δ (%) indicate the best solution value, the relative gap between our solution and the best solution provided in the literature (Desaulniers et al., 2016), the number of branches, the computation time in seconds, and the relative integrality gap, respectively. Moreover, Δ_{TPA} (%) under HCG denotes the relative gap between the bound obtained by HCG and that obtained by TPA. If TPA does not provide a lower bound within the run-time limit for an instance, the relative integrality gap cannot be calculated. Δ_{BKS} is not computed if the algorithm is not completed within the time limit. Additionally, Δ_{TPA} is computed only for the instances that could be solved by both algorithms.

Table A.1. MP problem results obtained by monodirectional algorithms for 25-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	62289	0	1.5	0.00	0.00	62289	0	1.3	0.00	0.00
C102	52169	0	8.38	0.00	0.00	52169	0	4.7	0.00	0.00
C103	34554	0	64.2	0.37	0.00	34554	0	25.7	0.00	0.00
C104	43192	0	79.7	0.00	0.11	43481	0	10.9	0.66	0.78
C105	52480	0	0.83	0.00	0.00	52480	0	0.7	0.00	0.00
C106	56002	0	1.25	1.93	0.72	56002	0	1.1	0.00	0.72
C107	50515	0	2.11	1.73	0.00	50515	0	1.9	0.00	0.00
C108	49482	0	1.95	0.00	0.00	49482	0	1.4	0.00	0.00
C109	44240	0	9.07	0.11	0.00	44654	0	4.6	0.93	0.94
RC101	72835	0	0.44	1.47	0.00	72835	0	0.4	0.00	0.00
RC102	63971	0	1.06	1.60	0.00	63971	0	0.8	0.00	0.00
RC103	55223	0	4.97	1.25	0.00	55223	0	3.4	0.00	0.00
RC104	51895	0	16.0	2.77	0.53	51895	0	7.4	0.00	0.53
RC105	58986	0	1.98	0.00	0.00	59037	0	1.0	0.09	0.09
RC106	54928	0	2.03	0.00	0.00	54979	0	1.3	0.09	0.09
RC107	49765	0	12.0	0.00	0.00	49765	0	4.6	0.00	0.00
RC108	47769	0	10.5	1.90	1.40	47769	0	5.5	0.00	1.40
R101	66146	0	0.17	0.00	0.00	66146	0	0.1	0.00	0.00
R102	43806	0	4.36	0.00	0.00	44048	0	1.5	0.55	0.55
R103	49310	0	6.33	0.24	0.00	49310	0	2.9	0.00	0.00
R104	35569	0	28.8	3.29	1.00	35569	0	14.6	0.00	1.00
R105	54407	0	0.78	0.01	0.00	54407	0	0.6	0.00	0.00
R106	48022	0	9.05	1.63	0.00	48022	0	6.0	0.00	0.00
R107	41114	0	69.9	0.52	0.00	41235	0	7.0	0.29	0.29
R108	42434	0	30.7	0.00	0.00	42548	0	8.9	0.27	0.27
R109	45351	0	2.8	1.61	0.00	45351	0	2.3	0.00	0.00
R110	40684	0	14.7	0.45	0.00	40684	0	10.1	0.00	0.00
R111	38736	0	18.7	4.01	1.17	38736	0	8.7	0.00	1.17
R112	38551	0	22.4	2.50	0.00	38551	0	10.6	0.00	0.00

Table A.2. MP problem results obtained by monodirectional algorithms for 50-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	77150	0	13.2	0.00	0.00	77150	0	11.6	0.00	0.00
C102	77719	0	107.2	0.00	0.00	77719	0	66.2	0.00	0.00
C103	63783	0	969.5	3.27	n/a	63783	0	208.9	0.00	n/a
C104						55099	0	1660.7	n/a	0.00
C105	72980	0	12.6	0.00	0.00	72980	0	10.9	0.00	0.00
C106	73554	0	17.3	0.06	0.00	73554	0	15.1	0.00	0.00
C107	70412	0	27.2	0.00	0.00	70412	0	24.4	0.00	0.00
C108	72984	0	51.2	2.00	1.52	72984	0	25.2	0.00	1.52
C109	66897	0	104.2	1.62	0.00	66944	0	50.6	0.07	0.07
RC101	105800	0	6.0	4.30	0.61	105800	0	5.3	0.00	0.61
RC102	87543	0	99.3	0.00	0.00	87543	0	65.4	0.00	0.00
RC103	82364	0	181.9	1.29	0.17	82364	0	73.7	0.00	0.17
RC104	69430	0	1110.4	2.62	1.07	69430	0	219.5	0.00	1.07
RC105	97125	0	20.5	2.84	0.43	97125	0	13.1	0.00	0.43
RC106	86817	0	42.8	3.17	1.12	86817	0	22.4	0.00	1.12
RC107	78368	0	180.8	4.11	n/a	78368	0	59.1	0.00	n/a
R101	93065	0	2.3	0.28	0.05	93065	0	1.9	0.00	0.05
R102	84919	0	44.1	1.39	0.05	84919	0	32.7	0.00	0.05
R103	77801	0	94.9	0.81	0.00	78089	0	37.6	0.37	0.37
R104	62199	0	2953.6	3.56	n/a	62199	0	314.3	0.00	n/a
R105	83260	0	9.8	1.75	0.38	83260	0	7.9	0.00	0.38
R106	78934	0	160.0	1.96	0.00	78934	0	55.0	0.00	0.00
R107	67580	0	372.8	1.88	0.95	67580	0	203.7	0.00	0.95
R109	76758	0	65.3	2.08	0.48	76789	0	29.7	0.04	0.52
R110	71318	0	310.6	3.31	1.23	71318	0	135.9	0.00	1.23
R111	72026	0	194.3	0.46	2.23	72026	0	97.1	0.00	0.20
R112	59617	0	1746.6	2.09	1.65	59617	0	237.7	0.00	1.65

Table A.3. MP problem results obtained by monodirectional algorithms for 100-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	104376	0	278.4	0.00	0.00	104376	0	260.4	0.00	0.00
C105	102786	0	612.8	4.57	1.15	102786	0	586.2	0.00	1.15
C106	102313	0	1638.5	4.46	1.46	102313	0	1567.6	0.00	1.46
RC101	163255	0	260.3	3.98	n/a	163255	0	77.7	0.00	n/a
RC102						151327	0	546.8	n/a	2.04
RC105	145921	0.0	2378.8	2.55	1.15	146849	0	165.7	n/a	1.79
R101	157560	0	44.5	0.62	0.11	159474	0	38.4	0.00	0.11
R102	142909	0	2199.6	0.66	0.34	143153	0	571.8	0.00	0.34
R105	131862	0	189.4	2.00	1.19	131862	0	147.7	0.00	1.19
R106						120760	0	2446.0	n/a	n/a
R109						118843	0	349.5	n/a	n/a
R110						109461	0	2655.5	n/a	n/a

Table A.4. SP problem results obtained by monodirectional algorithms for 25-customer instances

Ins.	TPA					HCG				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	62289	0	1.0	0.00	0.00	62289	0	0.8	0.00	0.00
C102	54130	0	4.0	0.00	0.00	54130	0	2.6	0.00	0.00
C103	34554	0	52.9	0.00	0.00	34554	0	11.9	0.00	0.00
C104	45968	0	16.6	0.00	0.00	45968	0	3.4	0.00	0.00
C105	53706	0	0.5	0.00	0.00	53706	0	0.4	0.00	0.00
C106	56002	0	0.6	1.93	0.42	56002	0	0.5	0.00	0.42
C107	50515	0	0.8	0.81	0.00	50515	0	0.7	0.00	0.00
C108	50915	0	1.2	0.00	0.00	50915	0	0.9	0.00	0.00
C109	49385	0	3.0	1.95	0.27	49516	0	2.4	0.00	0.27
RC101	76335	0	0.4	1.53	0.00	76335	0	0.3	0.00	0.00
RC102	67566	0	0.8	1.49	0.38	67825	0	0.6	0.00	0.38
RC103	55260	0	2.9	0.62	0.00	55223	0	1.9	0.00	0.00
RC104	52995	0	10.0	1.27	0.27	52062	0	3.1	0.00	0.27
RC105	59469	0	1.0	0.00	0.00	59469	0	0.6	0.00	0.00
RC106	55723	0	0.8	0.74	0.58	56047	0	0.6	0.00	0.58
RC107	50222	0	3.8	0.00	0.00	50222	0	2.5	0.00	0.00
RC108	47567	0	6.2	1.28	0.97	47567	0	3.9	0.00	0.97
R101	66230	0	0.1	0.00	0.00	66230	0	0.1	0.00	0.00
R102	45877	0	2.0	1.33	0.75	45692	0	0.8	0.00	0.75
R103	49858	0	2.5	0.00	0.00	49858	0	1.6	0.00	0.00
R104	35216	0	26.8	1.38	0.00	35216	0	6.2	0.00	0.00
R105	61638	0	0.4	0.72	0.00	61638	0	0.3	0.00	0.00
R106	48022	0	3.0	1.38	0.00	48022	0	2.0	0.00	0.00
R107	41651	0	22.2	1.07	0.01	41655	0	7.7	0.00	0.01
R108	45686	0	9.7	2.12	0.00	45652	0	5.0	0.00	0.00
R109	46203	0	1.6	0.38	0.00	46203	0	1.0	0.00	0.00
R110	41004	0	4.8	0.00	0.00	41004	0	2.9	0.00	0.00
R111	38304	0	16.9	1.76	0.38	38449	0	4.9	0.00	0.38
R112	38551	0	7.9	1.18	0.00	38551	0	3.0	0.00	0.00

Table A.5. SP problem results obtained by monodirectional algorithms for 50-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	77693	0	11.2	0.00	0.00	77828	0	7.7	0.17	0.17
C102	82333	0	121.9	0.00	0.00	82904	0	24.7	0.69	0.69
C103	67534	0	253.4	3.34	1.16	67534	0	50.6	0.00	1.16
C104						55913	0	280.3	n/a	0.00
C105	75563	0	6.9	0.61	0.12	77942	0	6.0	3.05	3.27
C106	79933	0	10.1	0.71	0.01	79933	0	8.9	0.00	0.01
C107	78729	0	10.5	4.58	0.56	79610	0	8.9	1.11	1.69
C108	73383	0	30.5	0.50	0.02	74596	0	13.5	1.63	1.67
C109	67740	0	37.5	1.58	0.35	67740	0	21.0	0.00	0.35
RC101	107507	0	3.4	3.02	0.81	107507	0	2.5	0.00	0.81
RC102	90071	0	33.7	0.27	0.00	90071	0	25.9	0.00	0.00
RC103	84700	0	110.6	0.65	0.00	85833	0	31.4	1.32	1.34
RC104	69294	0	440.2	1.99	0.13	69703	0	94.2	0.59	0.72
RC105	99759	0	15.7	1.11	0.00	101404	0	6.2	1.62	1.65
RC108	73602	0	296.5	3.34	1.37	74708	0	75.0	1.48	2.89
R101	93729	0	1.4	0.00	0.00	93729	0	1.2	0.00	0.00
R102	90035	0	15.4	3.01	2.50	90035	0	9.9	0.00	2.50
R103	80947	0	38.7	1.79	0.36	80947	0	17.0	0.00	0.36
R104	62561	0	1353.4	3.74	n/a	62561	0	165.1	0.00	n/a
R105	84055	0	5.2	1.57	0.19	84076	0	2.9	0.02	0.21
R106	80990	0	69.1	2.56	1.32	80990	0	13.5	0.00	1.32
R107	74229	0	154.4	2.64	0.63	74229	0	61.3	0.00	0.63
R108	53763	0	1349.4	2.18	0.00	54981	0	279.8	0.00	0.00
R109	82179	0	28.5	1.98	0.68	82179	0	10.3	0.00	0.68
R110	73971	0	172.5	2.84	1.41	73971	0	27.9	0.00	1.41
R111	79229	0	83.9	2.78	1.16	79420	0	25.1	0.24	1.41
R112	66054	0	832.0	4.85	n/a	66054	0	98.4	0.00	n/a

Table A.6. SP problem results obtained by monodirectional algorithms for 100-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	105706	0	315.1	0.00	0.00	105706	0	155.1	0.00	0.00
C102		0				102848	0	2081.1	n/a	0.30
C105	103250	0	661.9	1.06	0.12	103250	0	476.0	0.00	0.12
C106	102876	0	353.6	0.46	0.13	102876	0	310.3	0.00	0.13
C107	103256	0	442.9	1.92	0.21	103256	0	305.5	0.00	0.21
C108	102436	0	2068.0	2.86	0.94	102436	0	770.6	0.00	0.94
RC103						133355	0	1050.4	n/a	1.46
RC106	140885	0	1877.5	4.85	n/a	144388	0	106.3	2.43	n/a
R101	161850	0	29.6	1.57	0.95	161850	0	10.3	0.00	0.96
R102	145257	0	755.9	0.17	0.07	145257	0	253.1	0.00	0.07
R105	137557	0	69.0	2.67	n/a	137557	0	50.2	0.00	n/a
R106	125297	0	2004.3	1.09	0.09	125297	0	340.9	0.00	0.09
R107						118019	0	1276.9	n/a	n/a
R108						102222	0	2261.7	n/a	n/a
R111						112729	0	1294.1	n/a	n/a

Table A.7. MP problem results obtained by bidirectional algorithms for 25-customer instances

Ins.	TPA					HCG				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	62289	0	1.4	0.00	0.00	62289	0	1.1	0.00	0.00
C102	52169	0	10.3	0.00	0.00	52169	0	4.4	0.00	0.00
C103	34554	0	49.0	0.37	0.00	34554	0	11.7	0.00	0.00
C104	43143	0	249.8	0.00	0.00	43194	0	10.0	0.12	0.12
C105	52480	0	1.0	0.00	0.00	52480	0	0.8	0.00	0.00
C106	55851	0	1.5	1.67	0.45	55851	0	1.1	0.00	0.45
C107	50515	0	1.9	1.73	0.00	50515	0	1.4	0.00	0.00
C108	49482	0	2.1	0.00	0.00	49482	0	1.2	0.00	0.00
C109	44240	0	8.7	0.11	0.00	44498	0	3.4	0.58	0.58
RC101	72835	0	0.3	1.47	0.00	72835	0	0.2	0.00	0.00
RC102	63971	0	1.0	1.60	0.00	63971	0	0.8	0.00	0.00
RC103	55265	0	4.6	1.33	0.08	55265	0	2.6	0.00	0.08
RC104	51785	0	14.2	2.57	0.32	51785	0	6.6	0.00	0.32
RC105	58986	0	1.9	0.00	0.00	58986	0	1.2	0.00	0.00
RC106	54928	0	1.6	0.00	0.00	54979	0	1.0	0.09	0.09
RC107	49765	0	8.7	0.00	0.00	49995	0	2.4	0.46	0.46
RC108	47108	0	8.3	0.52	0.00	47108	0	5.2	0.00	0.00
R101	66146	0	0.2	0.00	0.00	66146	0	0.2	0.00	0.00
R102	43806	0	3.3	0.00	0.00	43806	0	2.2	0.00	0.00
R103	49310	0	6.0	0.24	0.00	49310	0	2.8	0.00	0.00
R104	35216	0	19.2	2.10	0.00	35276	0	12.8	0.17	0.17
R105	54407	0	0.7	0.01	0.00	54407	0	0.5	0.00	0.00
R106	48022	0	6.7	1.63	0.00	48022	0	5.2	0.00	0.00
R107	41114	0	29.0	0.52	0.00	41114	0	8.2	0.00	0.00
R108	42434	0	29.1	0.00	0.00	42548	0	6.5	0.27	0.27
R109	45660	0	4.1	2.27	0.68	45660	0	2.9	0.00	0.68
R110	40684	0	10.3	0.45	0.00	40684	0	8.3	0.00	0.00
R111	38470	0	19.6	3.34	0.47	38470	0	7.4	0.00	0.47
R112	38551	0	17.5	2.50	0.00	38551	0	8.3	0.00	0.00

Table A.8. MP problem results obtained by bidirectional algorithms for 50-customer instances

Ins.	TPA					HCG				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	77150	0	13.4	0.00	0.00	77236	0	7.0	0.11	0.11
C102	77719	0	110.4	0.00	0.00	77719	0	58.2	0.00	0.00
C103	63641	0	626.9	3.05	n/a	63718	0	147.4	0.12	n/a
C104	55099	0	1418.5	0.00	0.00	55239	0	310.5	0.25	0.25
C105	72980	0	20.5	0.00	0.00	72980	0	8.4	0.00	0.00
C106	73555	0	28.4	0.06	0.00	73554	0	15.4	0.00	0.00
C107	70412	0	34.3	0.00	0.00	70412	0	17.2	0.00	0.00
C108	74411	0	49.7	3.84	3.51	74077	0	21.1	-0.45	3.04
C109	67132	0	148.4	1.95	0.35	66899	0	32.4	-0.35	0.00
RC101	105681	0	7.4	4.19	0.50	105873	0	3.0	0.18	0.68
RC102	87543	0	82.7	0.00	0.00	87543	0	35.6	0.00	0.00
RC103	82221	0	84.6	1.12	0.00	82364	0	57.5	0.17	0.17
RC104	68933	0	424.3	1.92	0.34	69465	0	90.9	0.77	1.12
RC105	97193	0	24.2	2.90	0.50	97213	0	11.1	0.02	0.52
RC106	86995	0	34.4	3.37	1.33	88008	0	19.4	1.16	2.51
RC107	78867	0	148.5	4.72	n/a	78540	0	54.4	-0.41	n/a
RC108						73750	125	3181.0	n/a	2.32
R101	93065	0	3.1	0.28	0.05	93019	0	2.6	-0.05	0.00
R102	84877	0	36.0	1.35	0.00	85305	0	26.5	0.50	0.50
R103	78179	0	110.9	1.18	0.49	78089	0	33.0	-0.12	0.37
R104						62199	0	185.6	n/a	n/a
R105	83399	0	9.6	1.91	0.55	83438	0	7.1	0.05	0.60
R106	79009	0	81.4	1.95	0.10	79437	0	40.8	0.54	0.64
R107	66942	0	266.3	0.94	0.00	67778	0	158.3	1.25	1.25
R108	53624	0	1433.9	2.28	0.00	54173	0	175.0	1.02	1.02
R109	76538	0	54.8	1.80	0.19	76746	0	17.3	0.27	0.46
R110	70550	0	192.8	2.25	0.14	70777	0	65.8	0.32	0.46
R111	71880	0	195.3	0.26	0.00	71880	0	57.0	0.00	0.00
R112	59429	0	1255.3	1.77	1.33	60241	0	137.6	1.37	2.71

Table A.9. MP problem results obtained by bidirectional algorithms for 100-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	104376	0	306.4	0.00	0.00	104376	0	119.8	0.00	0.00
C102						101542	0	1461.5	n/a	n/a
C105	103017	0	889.5	4.79	1.38	102981	0	328.0	-0.03	1.34
C106	102223	0	1687.2	4.37	1.37	102699	0	635.1	0.47	1.84
C109						93688	0	1257.3	n/a	n/a
RC101	163949	0	260.4	4.39	n/a	164539	0	46.9	0.36	n/a
RC102	150185	0	1741.4	3.83	1.27	151424	0	364.0	0.82	2.10
RC105	146219	0	703.1	2.75	1.36	146219	0	294.0	0.00	1.36
R101	157407	0	34.1	0.48	0.02	157760	0	24.4	0.22	0.24
R102	142532	0	1685.8	0.39	0.07	142815	0	269.7	0.20	0.27
R103						117773	0	1012.9	n/a	1.30
R104						106742	0	2670.2	n/a	n/a
R105	132138	0	172.1	2.20	1.41	131119	0	97.9	-0.77	0.62
R106	120794	0	1552.0	0.48	n/a	120794	0	945.2	0.00	n/a
R107						112227	0	2327.3	n/a	n/a
R108						100652	0	2169.5	n/a	n/a
R109	118519	0	1921.2	3.08	n/a	119559	0	227.4	0.88	n/a
R110						109386	0	1474.5	n/a	n/a
R111						109820	0	1451.0	n/a	n/a

Table A.10. SP problem results obtained by bidirectional algorithms for 25-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	62289	0	0.9	0.00	0.00	62289	0	0.7	0.00	0.00
C102	54130	0	4.4	0.00	0.00	54130	0	1.5	0.00	0.00
C103	34554	0	44.7	0.00	0.00	34554	0	4.7	0.00	0.00
C104	45974	0	26.9	0.00	0.01	45974	0	3.6	0.00	0.01
C105	53706	0	0.4	0.00	0.00	53706	0	0.3	0.00	0.00
C106	56002	0	0.6	1.93	0.42	56002	0	0.5	0.00	0.42
C107	50515	0	0.8	0.81	0.00	50515	0	0.7	0.00	0.00
C108	50915	0	1.3	0.00	0.00	50915	0	0.9	0.00	0.00
C109	49566	0	2.7	2.04	0.37	49566	0	2.0	0.00	0.37
RC101	76335	0	0.3	1.53	0.00	76335	0	0.3	0.00	0.00
RC102	67566	0	0.9	1.11	0.00	67566	0	0.9	0.00	0.00
RC103	55223	0	3.5	0.62	0.00	55223	0	1.9	0.00	0.00
RC104	52062	0	9.6	1.27	0.27	52062	0	3.3	0.00	0.27
RC105	59469	0	1.0	0.00	0.00	59469	0	0.6	0.00	0.00
RC106	55723	0	0.8	0.17	0.00	55723	0	0.7	0.00	0.00
RC107	50222	0	2.7	0.00	0.00	50222	0	2.2	0.00	0.00
RC108	47665	0	6.2	1.49	1.18	47665	0	3.9	0.00	1.18
R101	66230	0	0.1	0.00	0.00	66230	0	0.1	0.00	0.00
R102	45692	0	0.7	1.13	0.75	45692	0	0.6	0.00	0.75
R103	49858	0	3.5	0.00	0.00	49858	0	1.5	0.00	0.00
R104	35216	0	28.1	1.38	0.00	35216	0	8.0	0.00	0.00
R105	61638	0	0.4	0.72	0.00	61638	0	0.3	0.00	0.00
R106	48022	0	3.1	1.38	0.00	48022	0	2.0	0.00	0.00
R107	41815	0	25.0	1.45	0.39	41815	0	5.8	0.00	0.39
R108	45650	0	12.5	2.12	0.00	45650	0	3.2	0.00	0.00
R109	47039	0	1.6	2.15	1.81	47039	0	0.8	0.00	1.81
R110	41004	0	4.1	0.00	0.00	41004	0	3.2	0.00	0.00
R111	38470	0	17.9	1.81	0.43	38470	0	4.8	0.00	0.43
R112	38551	0	8.4	1.18	0.00	38551	0	3.1	0.00	0.00

Table A.11. SP problem results obtained by bidirectional algorithms for 50-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	77693	0	8.5	0.00	0.00	77693	0	4.8	0.00	0.00
C102	82333	0	118.3	0.00	0.00	82333	0	19.9	0.00	0.00
C103	67521	0	298.0	3.32	1.14	67521	0	46.7	0.00	1.14
C104	55913	0	1378.2	0.00	0.00	55913	0	123.8	0.00	0.00
C105	75688	0	10.3	0.78	0.28	75688	0	5.5	0.00	0.28
C106	79933	0	8.9	0.71	0.01	79933	0	7.3	0.00	0.01
C107	78688	0	12.9	4.53	0.51	78688	0	9.3	0.00	0.51
C108	73383	0	29.5	0.50	0.02	74988	0	13.2	2.19	2.21
C109	67740	0	48.1	1.58	0.35	67740	0	18.7	0.00	0.35
RC101	107507	0	4.3	3.02	0.81	107507	0	2.0	0.00	0.81
RC102	90071	0	59.8	0.27	0.00	90071	0	15.2	0.00	0.00
RC103	84699	0	76.9	0.65	0.00	84699	0	35.6	0.00	0.00
RC104	69602	0	419.0	2.43	0.57	69602	0	84.4	0.00	0.57
RC105	99760	0	12.7	1.11	0.00	99957	0	5.8	0.20	0.20
RC108	73839	0	197.1	3.65	1.69	73961	0	70.5	0.17	1.86
R101	93729	0	2.4	0.00	0.00	93729	0	1.1	0.00	0.00
R102	90803	0	29.3	4.26	3.37	90803	0	9.3	0.00	3.37
R103	81269	0	28.0	2.18	0.76	81269	0	19.8	0.00	0.76
R104	63035	0	1674.4	4.46	n/a	63035	0	117.1	0.00	n/a
R105	84119	0	5.5	1.65	0.26	84119	0	4.1	0.00	0.26
R106	79974	0	60.3	1.60	0.05	79974	0	21.9	0.00	0.05
R107	74183	0	216.9	2.58	0.57	74184	0	58.4	0.00	0.57
R108	54726	0	1461.0	3.90	1.79	54908	0	119.4	0.33	2.13
R109	82424	0	28.0	2.33	0.98	82424	0	8.5	0.00	0.98
R110	73734	0	200.1	2.61	1.09	73734	0	27.8	0.00	1.09
R111	79811	0	74.5	3.54	1.91	79811	0	26.0	0.00	1.91
R112	66054	0	2115.4	4.85	n/a	66054	0	84.7	0.00	n/a

Table A.12. SP problem results obtained by bidirectional algorithms for 100-customer instances

Ins.	**TPA**					**HCG**				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ_{TPA} (%)	Δ_{BKS} (%)
C101	105706	0	215.9	0.00	0.00	105706	0	82.4	0.00	0.00
C102	102640	0	3230.9	0.85	0.10	102640	0	1329.7	0.00	0.10
C103						102411	0	2998.9	n/a	n/a
C105	103263	0	374.9	1.07	0.13	103368	0	167.9	0.10	0.24
C106	102744	0	803.9	0.33	0.00	103300	0	407.2	0.54	0.54
C107	103384	0	576.2	2.04	0.33	103328	0	278.7	-0.05	0.28
C108	102793	0	1456.5	3.20	1.29	102793	0	576.8	0.00	1.29
RC102	155304	0	1126.5	2.02	0.28	155055	0	383.0	-0.16	0.11
RC103	134046	0	3511.8	2.67	1.99	135699	0	828.2	1.23	n/a
RC104						121991	0	1743.9	n/a	n/a
RC106	140641	0	629.6	4.70	n/a	141040	0	260.3	0.28	n/a
R101	161515	0	64.8	1.37	0.75	160923	0	16.1	-0.37	0.38
R102	145156	0	1097.3	0.14	0.00	145211	0	191.7	0.04	0.04
R103						123315	0	571.5	n/a	1.84
R104						109146	0	1689.4	n/a	n/a
R105	138846	0	80.0	3.57	n/a	138846	0	46.5	0.00	n/a
R106	125456	0	1512.3	1.22	0.21	125456	0	365.6	0.00	0.21
R107						117628	0	1167.8	n/a	n/a
R108						102269	0	1192.4	n/a	n/a
R109	126350	0	1056.4	3.25	n/a	128072	0	162.6	1.36	n/a
R110						111691	0	668.5	n/a	n/a
R111						114357	0	924.2	n/a	n/a

Appendix B. Detailed Results of the Pulse Algorithm

This appendix provides the detailed results of the computational study conducted by using the Pulse algorithm with and without the augmented node (AN) approach for the multiple-partial recharge (MP) and the single-partial recharge (SP) problem variants over the complete data set. The columns Bound, Δ_{BKS} (%), #b, t(s), and Δ (%) indicate the best bound, the relative gap between our solution and the best solution provided in the literature (Desaulniers et al., 2016), the number of branches, the computation time in seconds, and the relative integrality gap, respectively.

Table B.1. MP problem results obtained for 25-customer instances

Ins.	$\mathbf{Pulse_0}$					$\mathbf{Pulse_{AN}}$				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)
C101	62289	0	3.1	0.00	0.00	62289	0	2.3	0	0.00
C102	52169	0	403.0	0.00	0.00	52169	0	49.9	0	0.00
C103	34845	0	605.0	1.20	0.84	34554	0	531.9	0	0.00
C104	43204	0	2899.5	0.00	0.14	43143	0	940.0	2.88	0.00
C105	53383	0	2.3	1.62	1.69	52642	0	1.7	2.7	0.31
C106	56044	0	6.0	2.00	0.79	56044	0	2.2	1.67	0.79
C107	50515	0	4.0	1.73	0.00	51008	0	2.2	1.46	0.97
C108	49482	0	3.7	0.00	0.00	49482	0	3.9	0	0.00
C109	44240	0	15.0	0.11	0.00	44240	0	13.9	0.11	0.00
RC101	74306	0	0.7	3.42	1.98	73614	0	0.4	1.45	1.06
RC102	63971	0	1.0	1.60	0.00	63971	0	0.8	1.13	0.00
RC103	55324	0	4.6	1.44	0.18	55324	0	3.1	1.44	0.18
RC104	51881	0	10.5	2.75	0.50	52437	0	13.6	1.95	1.55
RC105	58986	0	1.5	0.00	0.00	58986	0	1.8	0	0.00
RC106	54928	0	2.1	0.00	0.00	54928	0	1.5	0	0.00
RC107	49765	0	5.1	0.00	0.00	49765	0	5.9	0	0.00
RC108	47108	0	5.6	0.52	0.00	47318	0	6.8	0.32	0.44
R101	66149	0	0.5	0.00	0.00	66149	0	0.3	0	0.00
R102	43806	0	3.2	0.00	0.00	43806	0	1.7	0	0.00
R103	49310	0	2.3	0.24	0.00	49310	0	2.9	0.43	0.00
R104	35455	0	17.8	2.98	0.67	35216	0	12.9	1.46	0.00
R105	54407	0	0.9	0.01	0.00	54407	0	0.7	0.04	0.00
R106	48078	0	4.2	1.74	0.12	48334	0	4.0	1.38	0.65
R107	41227	0	31.7	0.80	0.27	41192	0	13.3	0.71	0.19
R108	42434	0	25.6	0.00	0.00	42434	0	13.3	0	0.00
R109	45351	0	3.2	1.60	0.00	45351	0	2.1	1.45	0.00
R110	40684	0	8.5	0.45	0.00	40904	0	5.3	0.45	0.54
R111	38364	0	50.2	3.07	0.20	38500	0	28.2	1.99	0.55
R112	38551	0	12.5	2.50	0.00	38776	0	12.8	2.4	0.58

Table B.2. MP problem results obtained for 50-customer instances

Ins.	$\mathbf{Pulse_0}$					$\mathbf{Pulse_{AN}}$				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)
C101	77150	0	17.7	0.00	0.00	77150	0	19.4	0.00	0.00
C102	77719	0	918.2	0.00	0.00	77719	0	516.4	0.00	0.00
C103	64403	0	590.2	4.06	n/a	64074	0	1262.6	3.62	n/a
C105	72980	0	23.2	0.00	0.00	72980	0	38.6	0.00	0.00
C106	73555	0	40.2	0.06	0.00	73554	0	35.5	0.06	0.00
C107	70412	0	57.4	0.00	0.00	70412	0	42.4	0.00	0.00
C108	72067	0	63.3	0.53	0.24	73132	0	48.9	1.86	1.70
C109	66916	0	129.0	1.37	0.03	67141	0	104.8	1.58	0.36
RC102	87662	0	21.4	0.00	0.14	87543	0	37.6	0.00	0.00
RC103	82314	0	63.0	1.21	0.11	82927	0	124.1	1.17	0.85
RC104	69108	0	183.2	2.17	0.59	68934	0	329.7	1.88	0.34
RC105	97213	0	9.5	2.92	0.52	97038	0	13.0	2.75	0.34
RC106	87409	0	14.8	3.83	1.78	87998	0	14.9	4.47	2.44
RC108	72976	0	121.5	4.28	1.23	75342	844	3615.4	6.42	4.33
R101	93144	0	5.0	0.37	0.13	93019	0	4.7	0.23	0.00
R102	86209	0	12.4	2.87	1.55	85463	0	17.4	2.02	0.69
R103	77989	0	23.3	0.97	0.24	77933	0	39.0	0.67	0.17
R104	62779	0	239.0	4.45	n/a	62789	0	1397.6	4.20	n/a
R105	83477	0	10.0	1.87	0.64	83416	0	6.7	1.83	0.57
R106	79922	0	36.5	3.17	1.24	79884	0	85.4	2.95	1.19
R107	68515	0	84.4	3.22	2.30	66982	0	118.0	0.98	0.06
R108	55191	368	3601.9	5.0	2.84	53763	0	577.8	2.15	0.26
R109	77551	0	18.1	3.07	1.49	78533	0	20.1	3.93	2.73
R110	70892	0	62.4	2.71	0.62	73826	1042	3611.3	5.55	4.57
R111	71883	0	39.3	0.19	0.00	75012	0	66.0	3.39	4.18
R112	60374	0	934.2	3.42	2.85	59843	0	3419.8	1.54	1.99

Table B.3. MP problem results obtained for 100-customer instances

Ins.	$\mathbf{Pulse_0}$					$\mathbf{Pulse_{AN}}$				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)
C101	104376	0	401.4	0.00	0.00	104376	0	689.0	0.00	0.00
C105	102950	0	569.0	4.71	1.30	102660	0	2146.2	4.43	1.02
C106	102524	0	1379.9	4.64	1.64	102586	23	3037.3	4.68	1.70
RC101	163757	0	63.1	4.25	n/a	165983	662	3667.7	5.35	n/a
RC102	152146	0	214.3	4.97	2.52	151996	0	676.2	4.53	2.43
RC105	147203	0	121.0	3.40	2.00	146649	0	320.7	2.74	1.63
R101	158018	0	50.9	0.91	0.40	157527	0	47.7	0.60	0.09
R102	142787	0	265.6	0.57	0.25	142886	0	752.2	0.61	0.32
R103	116824	0	1024.5	1.01	0.48	205734	0	3700.6	0.00	n/a
R105	133872	0	102.0	3.45	2.66	133395	0	69.8	2.68	2.31
R106	120959	0	644.5	0.54	n/a	122919	0	2982.2	1.55	n/a
R107	113925	0	809.5	4.33	n/a	198226	0	3683.5	0.00	n/a
R109	119497	0	275.8	3.75	n/a	120200	0	881.5	4.14	n/a
R111	110144	0	954.8	4.95	n/a	201472	0	3638.6	0.00	n/a

Table B.4. SP problem results obtained for 25-customer instances

Ins.	$\mathbf{Pulse_0}$					$\mathbf{Pulse_{AN}}$				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)
C102	54130	0	93.0	0.00	0.00	54130	0	32.8	0.00	0.00
C103	34554	0	109.0	0.00	0.00	34554	0	220.1	0.00	0.00
C104	45968	0	950.2	0.00	0.00	45968	0	522.1	0.00	0.00
C105	53706	0	2.5	0.00	0.00	53706	0	1.3	0.00	0.00
C106	56002	0	2.6	1.93	0.42	56002	0	1.1	1.93	0.42
C107	50515	0	3.7	0.72	0.00	50515	0	2.1	0.72	0.00
C108	50915	0	4.7	0.00	0.00	50915	0	3.0	0.00	0.00
C109	49385	0	30.7	1.69	0.00	49385	0	18.1	1.70	0.00
RC101	76335	0	0.6	1.53	0.00	76335	0	0.4	1.53	0.00
RC102	67566	0	1.1	1.11	0.00	67566	0	0.5	1.11	0.00
RC103	55260	0	2.0	0.69	0.07	55223	0	2.7	0.62	0.00
RC104	52628	0	7.4	2.33	1.35	52566	0	10.3	2.21	1.23
RC105	59469	0	0.8	0.00	0.00	59469	0	1.0	0.00	0.00
RC106	56352	0	1.2	1.28	1.12	55723	0	1.2	0.17	0.00
RC107	50222	0	7.1	0.00	0.00	50222	0	4.3	0.00	0.00
RC108	48083	0	5.3	2.34	2.03	47108	0	4.6	0.32	0.00
R101	66230	0	0.4	0.00	0.00	66230	0	0.3	0.00	0.00
R102	46274	0	1.2	2.57	2.00	45350	0	1.2	0.59	0.00
R103	49858	0	1.6	0.00	0.00	49858	0	2.0	0.00	0.00
R104	35216	0	9.2	1.38	0.00	35216	0	8.2	1.38	0.00
R105	61638	0	0.8	0.72	0.00	61638	0	0.5	0.72	0.00
R106	48022	0	2.4	1.38	0.00	48022	0	3.0	1.38	0.00
R107	41711	0	9.2	1.21	0.14	41711	0	9.6	1.21	0.14
R108	45863	0	8.5	2.58	0.46	45650	0	6.9	2.12	0.00
R109	47144	0	1.8	2.37	2.00	46870	0	1.4	1.77	1.42
R110	41004	0	6.0	0.00	0.00	41004	0	5.4	0.00	0.00
R111	38470	0	17.5	1.81	0.43	38304	0	13.2	1.38	0.00
R112	38551	0	6.6	1.18	0.00	38551	0	6.8	1.18	0.00

Table B.5. SP problem results obtained for 50-customer instances

Ins.	$\mathbf{Pulse_0}$					$\mathbf{Pulse_{AN}}$				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)
C101	77693	0	25.4	0.00	0.00	77693	0	15.0	0.00	0.00
C102	82333	0	928.8	0.00	0.00	82333	0	455.4	0.00	0.00
C103	67600	0	840.1	3.27	1.24	67492	0	893.4	3.11	1.08
C105	75563	0	19.3	0.61	0.12	75563	0	21.1	0.61	0.12
C106	79933	0	46.1	0.67	0.01	79933	0	33.2	0.71	0.01
C107	78663	0	50.4	4.48	0.47	79190	1320	3611.9	5.12	1.14
C108	73383	0	79.8	0.50	0.02	73551	0	44.1	0.72	0.25
C109	69168	0	255.0	3.21	2.40	68951	0	77.7	3.09	2.09
RC101	107921	0	6.4	3.39	1.18	108278	0	4.0	3.71	1.51
RC102	90169	0	13.3	0.38	0.11	90071	0	18.7	0.27	0.00
RC103	86115	0	47.6	2.28	1.64	85103	0	115.6	1.12	0.47
RC104	69604	0	94.9	2.43	0.57	69205	0	222.1	1.87	0.00
RC105	100078	0	12.4	1.43	0.32	101043	0	12.5	2.37	1.27
RC108	74910	0	68.0	5.00	3.07	74295	0	195.0	4.21	2.27
R101	93729	0	3.0	0.00	0.00	93729	0	2.0	0.00	0.00
R102	90159	0	10.0	3.57	2.57	90042	0	14.0	3.45	2.45
R103	81305	0	12.9	2.22	0.80	81218	0	25.0	2.12	0.70
R104	63320	392	3168.3	4.86	n/a	62568	0	992.0	3.71	n/a
R105	83963	0	6.8	1.31	0.08	84139	0	4.0	1.41	0.29
R106	80821	0	21.6	2.63	1.09	81247	0	39.0	3.14	1.61
R107	74315	0	72.6	2.75	0.74	74106	0	93.0	2.48	0.46
R108	54137	0	507.1	2.78	0.69	53763	0	440.0	2.15	0.00
R109	83842	0	22.0	3.98	2.65	82483	0	21.0	2.40	1.05
R110	75448	0	39.8	4.73	3.32	75561	0	118.0	4.96	3.47
R111	81838	1350	3600.2	5.63	4.30	81242	0	65.0	4.94	3.60
R112	85910	n/a	3749.6	n/a	n/a	66085	0	949.0	4.90	n/a

Table B.6. SP problem results obtained for 100-customer instances

Ins.	$\mathbf{Pulse_0}$					$\mathbf{Pulse_{AN}}$				
	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)	Bound	#b	t (s)	Δ (%)	Δ_{BKS} (%)
C101	105706	0	640.2	0.00	0.00	105706	0	483.0	0.00	0.00
C105	103158	0	992.1	0.97	0.03	103131	0	1158.0	0.94	0.01
C106	177588	0	3602.6	n/a	n/a	102831	0	2718.0	0.41	n/a
C107	103294	0	1770.1	1.95	0.25	103530	0	1970.6	2.16	0.48
RC101	173326	0	68.2	4.45	n/a	174222	0	43.0	4.94	n/a
RC102	157280	0	429.5	3.27	1.55	156778	0	925.0	2.96	1.23
RC106	140743	0	313.8	4.77	n/a	140022	0	363.1	4.14	n/a
R101	161576	0	28.4	1.40	0.79	161526	0	30.0	1.37	0.76
R102	146004	0	279.7	0.72	0.58	145996	0	588.0	0.72	0.58
R103	124354	0	1712.9	4.09	2.70	205734	0	3630.0	0.00	n/a
R105	139670	0	100.4	4.12	n/a	141197	562	3612.0	5.14	n/a
R106	126180	0	735.1	1.77	0.79	125517	0	1964.9	1.26	0.26

BIBLIOGRAPHY

Ahmad, F., Alam, M. S., Alsaidan, I. S., and Shariff, S. M. (2020). Battery swapping station for electric vehicles: opportunities and challenges. *IET Smart Grid*, 3(3):280–286.

Andelmin, J. and Bartolini, E. (2017). An exact algorithm for the green vehicle routing problem. *Transportation Science*, 51(4):1288–1303.

Artmeier, A., Haselmayr, J., Leucker, M., and Sachenbacher, M. (2010). The shortest path problem revisited: Optimal routing for electric vehicles. In *Annual Conference on Artificial Intelligence*, pages 309–316. Springer.

Balcik, B. (2017). Site selection and vehicle routing for post-disaster rapid needs assessment. *Transportation research part E: logistics and transportation review*, 101:30–58.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations research*, 61(2):298–314.

Beheshti, A. K., Hejazi, S. R., and Alinaghian, M. (2015). The vehicle routing problem with multiple prioritized time windows: A case study. *Computers & Industrial Engineering*, 90:402–413.

Bektaş, T. and Laporte, G. (2011). The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250.

Belhaiza, S., Hansen, P., and Laporte, G. (2014). A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269–281.

Bettinelli, A., Ceselli, A., and Righini, G. (2014). A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. *Mathematical Programming Computation*, 6(2):171–197.

Boland, N., Dethridge, J., and Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58–68.

Breunig, U., Baldacci, R., Hartl, R. F., and Vidal, T. (2019). The electric two-echelon vehicle routing problem. *Computers & Operations Research*, 103:198–210.

Bruglieri, M., Mancini, S., Pezzella, F., and Pisacane, O. (2016). A new mathematical programming model for the green vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 55:89–92.

Bruglieri, M., Pezzella, F., Pisacane, O., Suraci, S., et al. (2015). A variable neighborhood search branching for the electric vehicle routing problem with time windows. *Electronic Notes in Discrete Mathematics*, 47:221–228.

Cabrera, N., Medaglia, A. L., Lozano, L., and Duque, D. (2020). An exact bidirectional pulse algorithm for the constrained shortest path. *Networks*, 76(2):128–146.

Çatay, B. and Keskin, M. (2017). The impact of quick charging stations on the route planning of electric vehicles. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 152–157. IEEE.

Coelho, L. C. and Laporte, G. (2013). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, 51(23-24):7156–7169.

Conrad, R. G. and Figliozzi, M. A. (2011). The recharging vehicle routing problem. In *Proceedings of the 2011 Industrial Engineering Research Conference*, page 8. IISE Norcross, GA.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.-S. (2005). New heuristics for the vehicle routing problem. *Logistics systems: design and optimization*, pages 279–297.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46.

Danna, E. and Pape, C. L. (2005). Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In *Column generation*, pages 99–129. Springer.

de Jong, C., Kant, G., and Van Vlient, A. (1996). On finding minimal route duration in the vehicle routing problem with multiple time windows. *Manuscript, Department of Computer Science, Utrecht University, Holland*.

Dell'Amico, M., Monaci, M., Pagani, C., and Vigo, D. (2007). Heuristic approaches for the fleet size and mix vehicle routing problem with time windows. *Transportation Science*, 41(4):516–526.

Dell'Amico, M., Righini, G., and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation science*, 40(2):235–247.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405.

Desaulniers, G., Lessard, F., and Hadjar, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404.

Desaulniers, G., Rakke, J. G., and Coelho, L. C. (2015). A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50(3):1060–1076.

DHL (2021). Winning logistics strategies for the last mile. https://www.dhl.com/global-en/home/insights-and-innovation/insights/shortening-the-last-mile.html. (last accessed: 2021-11-18).

Di Puglia Pugliese, L. and Guerriero, F. (2012). A computational study of solution approaches for the resource constrained elementary shortest path problem. *Annals of Operations Research*, 201(1):131–157.

Digital Commerce (2021). 360 analysis of U.S commerce department data. https://www.digitalcommerce360.com/article/e-commerce-sales-retail-sales-ten-year-review. (last accessed: 2021-11-18).

Doerner, K. F., Gronalt, M., Hartl, R. F., Kiechle, G., and Reimann, M. (2008). Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. *Computers & Operations Research*, 35(9):3034–3048.

Duman, E. N., Taş, D., and Çatay, B. (2021). Branch-and-price-and-cut methods for the electric vehicle routing problem with time windows. *International Journal of Production Research*, pages 1–22.

EC (2020). European Cmission Mobility Transport European Sustainable and Smart Mobility Strategy. https://ec.europa.eu/transport/themes/mobilitystrategy_en.

Elshaer, R. and Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers Industrial Engineering*, 140:106242.

eMarketer (2021). Retail ecommerce performance metrics. https://www.emarketer.com/forecasts/58fe47a2d2670009840a9ec7/58dd63dd2357af0c900b4d33. (last accessed: 2021-11-18).

Erdoğan, S. and Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114.

Favaretto, D., Moretti, E., and Pellegrini, P. (2007). Ant colony system for a vrp with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2):263–284.

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

Felipe, Á., Ortuño, M. T., Righini, G., and Tirado, G. (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111–128.

Ganesh, K. and Narendran, T. (2007). Cloves: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. *European Journal of Operational Research*, 178(3):699–717.

Giordano, A., Fischbeck, P., and Matthews, H. S. (2018). Environmental and economic comparison of diesel and battery electric delivery vans to inform city logistics fleet replacement strategies. *Transportation Research Part D: Transport and Environment*, 64:216–229.

Goeke, D. (2019). Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research*, 278(3):821–836.

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

Grønhaug, R., Christiansen, M., Desaulniers, G., and Desrosiers, J. (2010). A branch-and-price method for a liquefied natural gas inventory routing problem. *Transportation Science*, 44(3):400–415.

Guarnieri, M. (2012). Looking back to electric cars. In *2012 Third IEEE HISTory of ELectro-technology CONference (HISTELCON)*, pages 1–6. IEEE.

Hiermann, G., Puchinger, J., Ropke, S., and Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018.

Ho, S. C., Szeto, W. Y., Kuo, Y.-H., Leung, J. M., Petering, M., and Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421.

Hof, J., Schneider, M., and Goeke, D. (2017). Solving the battery swap station location-routing problem with capacitated electric vehicles using an avns algorithm for vehicle-routing problems with intermediate stops. *Transportation Research Part B: Methodological*, 97:102–112.

IBM (2020). ILOG CPLEX Optimizer 20.1.0. https://www.ibm.com/tr-tr/products/ilog-cplex-optimization-studio.

IEA (2019). Transport sector CO2 emission by mode in the Sustainable Development Scenario, 2000-2030. https://www.iea.org/data-and-statistics/charts/transport-sector-co2-emissions-by-mode-in-the-sustainable-development-scenario-2000-2030. (last accessed: 2021-04-20).

IEA (2020). International Energy Agency, Clean Energy Ministerial, and Electric Vehicles Initiative (EVI), (June 2020), "Global EV Outlook 2020: Entering the decade of electric drive?". https://doi.org/10.1787/d394399e-en. (last accessed: 2021-06-15).

IntelliJ Idea (2020). Jet Beans. https://www.jetbrains.com/idea/.

Jaller, M., Pineda, L., and Ambrose, H. (2018). Evaluating the use of zero-emission vehicles in last mile deliveries. *ITS Reports*, 2017(33).

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

Jie, W., Yang, J., Zhang, M., and Huang, Y. (2019). The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. *European Journal of Operational Research*, 272(3):879–904.

Kallehauge, B., Larsen, J., Madsen, O. B., and Solomon, M. M. (2005). Vehicle routing problem with time windows. In *Column Generation*, pages 67–98. Springer.

Keskin, M., Akhavan-Tabatabaei, R., and Çatay, B. (2019a). Electric vehicle routing problem with time windows and stochastic waiting times at recharging stations. In *2019 Winter Simulation Conference (WSC)*, pages 1649–1659. IEEE.

Keskin, M. and Çatay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111–127.

Keskin, M. and Çatay, B. (2018). A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172–188.

Keskin, M., Çatay, B., and Laporte, G. (2021). A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations. *Computers & Operations Research*, 125:105060.

Keskin, M., Laporte, G., and Çatay, B. (2019b). Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research*, 107:77–94.

Kohl, N., Desrosiers, J., Madsen, O. B., Solomon, M. M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116.

Kullman, N., Goodson, J., and Mendoza, J. E. (2018). Dynamic electric vehicle routing with mid-route recharging and uncertain availability. In *ODYSSEUS 2018*.

Li, B., Xu, S., and Peng, H. (2020). Eco-routing for plug-in hybrid electric vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE.

Lozano, L., Duque, D., and Medaglia, A. L. (2015). An exact algorithm for the elementary shortest path problem with resource constraints. *Transportation Science*, 50(1):348–357.

Luby, M. and Ragde, P. (1989). A bidirectional shortest-path algorithm with good average-case behavior. *Algorithmica*, 4(1):551–567.

Macrina, G., Pugliese, L. D. P., Guerriero, F., and Laporte, G. (2019). The green mixed fleet vehicle routing problem with partial battery recharging and time windows. *Computers & Operations Research*, 101:183–199.

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

Mangiaracina, R., Marchet, G., Perotti, S., and Tumino, A. (2015). A review of the environmental implications of b2c e-commerce: a logistics perspective. *International Journal of Physical Distribution & Logistics Management*.

Mercedes-Benz (2021). EQV. https://ev-database.org/car/1240/Mercedes-EQV-300-Long. (last accessed: 2021-11-27).

Molenbruch, Y., Braekers, K., and Caris, A. (2017). Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1):295–325.

Montoya, A., Guéret, C., Mendoza, J. E., and Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110.

Muller, S. J. W. (2022). How do we get the adoption of electric vehicles into a higher gear in europe?

Nykvist, B. and Nilsson, M. (2015). Rapidly falling costs of battery packs for electric vehicles. *Nature climate change*, 5(4):329–332.

Ozbaygin, G., Karasan, O. E., Savelsbergh, M., and Yaman, H. (2017). A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 100:115–137.

Paz, J., Granada-Echeverri, M., and Escobar, J. (2018). The multi-depot electric vehicle location routing problem with time windows. *International Journal of Industrial Engineering Computations*, 9(1):123–136.

Pelletier, S., Jabali, O., and Laporte, G. (2019). The electric vehicle routing problem with energy consumption uncertainty. *Transportation Research Part B: Methodological*, 126:225–255.

Pelletier, S., Jabali, O., Laporte, G., and Veneroni, M. (2017). Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. *Transportation Research Part B: Methodological*, 103:158–187.

Petch, R. J. and Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133(1-3):69–92.

Petersen, B., Pisinger, D., and Spoorendonk, S. (2008). Chvátal-gomory rank-1 cuts used in a dantzig-wolfe decomposition of the vehicle routing problem with time windows. In *The vehicle routing problem: latest advances and new challenges*, pages 397–419. Springer.

Pistoia, G. (2010). *Electric and hybrid vehicles: Power sources, models, sustainability, infrastructure and the market*. Elsevier.

Pohl, I. (1971). Bi-directional search. *Machine intelligence*, 6:127–140.

Qin, H., Su, X., Ren, T., and Luo, Z. (2021). A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management*, 8(3):370–389.

Reyes, D., Savelsbergh, M., and Toriello, A. (2017). Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, 80:71–91.

Righini, G. and Salani, M. (2006). Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks: An International Journal*, 51(3):155–170.

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2016). Branch-and-price-and-cut for a service network design and hub location problem. *European Journal of Operational Research*, 255(3):935–947.

Sadati, M. E. H., Akbari, V., and Çatay, B. (2022). Electric vehicle routing problem with flexible deliveries. *International Journal of Production Research*, pages 1–27.

Santos, F. A., Mateus, G. R., and Salles da Cunha, A. (2011). A novel column generation algorithm for the vehicle routing problem with cross-docking. In *International Conference on Network Optimization*, pages 412–425. Springer.

Sassi, O. and Oulamara, A. (2017). Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–535.

Schiffer, M. and Walther, G. (2017a). An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Science*, 52(2):331–352.

Schiffer, M. and Walther, G. (2017b). The electric location routing problem with time windows and partial recharging. *European Journal of Operational Research*, 260(3):995–1013.

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

Semet, F., Toth, P., and Vigo, D. (2014). Chapter 2: Classical exact algorithms for the capacitated vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 37–57. SIAM.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Taş, D. (2021). Electric vehicle routing with flexible time windows: a column generation solution approach. *Transportation Letters*, 13(2):97–103.

Taş, D., Gendreau, M., Dellaert, N., Van Woensel, T., and De Kok, A. (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3):789–799.

Thomas, B. W., Calogiuri, T., and Hewitt, M. (2019). An exact bidirectional a approach for solving resource-constrained shortest path problems. *Networks*, 73(2):187–205.

Tilk, C. and Goel, A. (2020). Bidirectional labeling for solving vehicle routing and truck driver scheduling problems. *European Journal of Operational Research*, 283(1):108–124.

Tilk, C., Olkis, K., and Irnich, S. (2021). The last-mile vehicle routing problem with delivery options. *OR Spectrum*, pages 1–28.

Vallera, A., Nunes, P., and Brito, M. (2021). Why we need battery swapping technology. *Energy Policy*, 157:112481.

Vincent, F. Y., Redi, A. P., Hidayat, Y. A., and Wibowo, O. J. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing*, 53:119–132.

Wu, X., Freese, D., Cabrera, A., and Kitch, W. A. (2015). Electric vehicles' energy consumption measurement and estimation. *Transportation Research Part D: Transport and Environment*, 34:52–67.

Xu, D., Li, K., Zou, X., and Liu, L. (2017). An unpaired pickup and delivery vehicle routing problem with multi-visit. *Transportation Research Part E: Logistics and Transportation Review*, 103:218–247.

Yang, J. and Sun, H. (2015). Battery swap station location-routing problem with capacitated electric vehicles. *Computers & Operations Research*, 55:217–232.

Yassen, E. T., Ayob, M., Nazri, M. Z. A., and Sabar, N. R. (2017). An adaptive hybrid algorithm for vehicle routing problems with time windows. *Computers & Industrial Engineering*, 113:382–391.

Yoshio, M., Brodd, R. J., and Kozawa, A. (2009). *Lithium-ion batteries*, volume 1. Springer.

Yu, M., Nagarajan, V., and Shen, S. (2022). Improving column generation for vehicle routing problems via random coloring and parallelization. *INFORMS Journal on Computing*, 34(2):953–973.

Yuan, B., Liu, R., and Jiang, Z. (2015). A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. *International Journal of Production Research*, 53(24):7450–7464.

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

Zhen, L., Xu, Z., Ma, C., and Xiao, L. (2020). Hybrid electric vehicle routing problem with mode selection. *International Journal of Production Research*, 58(2):562–576.

Zhu, L. and Hu, D. (2019). Study on the vehicle routing problem considering congestion and emission factors. *International Journal of Production Research*, 57(19):6115–6129.

Zivin, J. S. G., Kotchen, M. J., and Mansur, E. T. (2014). Spatial and temporal heterogeneity of marginal emissions: Implications for electric cars and other electricity-shifting policies. *Journal of Economic Behavior & Organization*, 107:248–268.