Thèse n° 7431

The electric autonomous dial-a-ride problem

Présentée le 25 septembre 2020

à la Faculté de l'environnement naturel, architectural et construit
Laboratoire de systèmes de transports urbains
Programme doctoral en génie civil et environnement

pour l'obtention du grade de Docteur ès Sciences

par

Claudia BONGIOVANNI

Acceptée sur proposition du jury

Prof. A. M. Alahi, président du jury
Prof. N. Geroliminis, Dr M. Kaspi, directeurs de thèse
Prof. J.-F. Cordeau, rapporteur
Prof. K. Braekers, rapporteur
Prof. M. Bierlaire, rapporteur

École
polytechnique
fédérale
de Lausanne

2020

“Ero intero e tutte le cose erano per me naturali e confuse stupide come laria; credevo di veder tutto e non era che la scorza. Se mai tu diventerai metà di te stesso e te lauguro ragazzo capirai cose al di là della comune intelligenza dei cervelli interi. Avrai perso metà di te e del mondo ma la metà rimasta sarà mille volte più profonda e preziosa. E tu pure vorrai che tutto sia dimezzato e straziato a tua immagine perché bellezza e sapienza e giustizia ci sono solo in ciò che è fatto a brani."

Il visconte dimezzato,
Italo Calvino

“I was whole and all things were natural and confused to me, as stupid as air; I thought I saw everything and it was only the peel. If you ever become half of yourself, and I wish it upon you, son, you will understand things beyond the common intelligence of whole brains. You will have lost half of yourself and the world but the remaining half will be thousand times deeper and more precious. And you too will want everything to be halved and torn in your image, because beauty and wisdom and justice are only in what is made in pieces."

The cloven viscount,
Italo Calvino

To my family

Acknowledgements

When I moved to Switzerland five years ago I was extremely impressed by the positive and friendly atmosphere at the transportation laboratories at EPFL. The Urban Transport System Laboratory (LUTS) especially impressed me for its diversity, both at a technical and personal level. As such, I would like to first and foremost thank my PhD thesis Advisor, Prof. Nikolas Geroliminis, for welcolming and giving me the opportunity to be part of LUTS. Efharistó polí Nikola, for your trust and encouragement during my whole PhD. Secondly, I would like to thank my thesis co-Advisor, Dr Mor Kaspi, for guiding me through the most detailed and cumbersome aspects of my thesis. Toda raba Mor, our discussions, even the most animated, will be part of the best memories from my PhD. I would also like to express my sincere gratitude to the thesis jury members Prof. Michel Bierlaire, Prof. Kris Braekers, Prof. Jean-François Cordeau and Prof. Alexandre Alahi for their thorough revision of my work, as well as for all the fine informal talks. I especially thank Jean-François for also giving me the opportunity to collaborate and visit him at the CIRRELT in Montréal, it was a great experience.

My thesis was also composed of numerous lucky encounters which greatly enriched my experience at EPFL. Thank you Christine for your immense support to all LUTS members, for your grace, wisdom, empathy and sensitivity. Thank you LUTS friends Tasos (a.k.a. “Taso mou"), Reza, Mehmet, Nan, Mikhail, Raphaël, Isik (a.k.a. “big boss"), Leo (a.k.a. “beddazzo"), Martin (a.k.a. “poulet"), Semin (for our numerous talks about Korean movies), Dimitris (also a.k.a. “Dimitri mou"), Manos, Caio, Patrick, Martí, Anna, Sohyeong, and Ortal. Thank you TRANSP-OR friends Mila, Iliya (thanks for teaching me insightful Bulgarian terms), Eva (a.k.a. “the Greek goddess"), Marija (živeli!), Matthieu (a.k.a. “monsieur le professeur"), Shadi, Yousef, Anna, Stefan, Tom, Riccardo, Stefano Moret, Meri, Virginie, Gael (a.k.a. “trois fûts"), Nicholas (also a.k.a. “trois fûts"), Tim (thank you for unsuccessfully trying to teach me British tongue-twisters), Stefano Bortolomiol, Janody, Selin, and Nicola. Thank you VITA friends and most especially Alexandre, George (a.k.a. “Georgissimo"), and Lorenzo. Thank you CIRRELT friends Maëlle, Hani, Julien, and Larissa. Thank you LESO-PB friends Anna (a.k.a. “Anna love"), Marta, Alina, Pietro, Olivia, Luc, Jérémy, Andreas, Roberto, and Dan.

My thesis built on a solid foundation of friends that I encountered during my path from Rome to Berkeley and Lausanne. They were an important part of my life before, during, and certainly after the PhD. Thanks to my very best friends Ariel, Stefano, and Marco (a.k.a. “pallini"). Thank you to all of my friends from Rome and most especially Silvia, Marta, Flavia, Carlotta, Maggie, Fede, Claudio, Marco, Luca, Leonardo, Lofi, Valerio, Spezza, Vincenzo, Mucca, and Pollo. Thank you to my very good friends from Berkeley and most especially to Marco, Giangiacomo, and Massimo. Thanks to all my friends from Lausanne and most especially Thomas, Cassandre, Marc, Boris, and Quentin. Thank you to all my friends from Paris, most especially Tarak and Camille.

Very special thanks go to my previously mentioned friend and partner Dan. Merci Tatti, Boulo, Bibi, Bouzz... The many nicknames I have come up with for you will never be sufficient to express my gratitude for our first encounter and life together thereafter. Finally the deepest thanks go to my family. Thank you Mada, Dado and Giuz for everything I was, I am, and I will be, together. Thank you little Flavio for recently joining our family, you give me a glimpse into the future. Thank you zio Alberto, zia Betta and Chiara for being always there for me, you have my unconditional love. Finally, thank you brave aviator and gracious adventurer always in my heart, nonno Costantino and nonna Adriana, you have my forever love.

Ad maiora!

Lausanne, 25.06.2020 Claudia

Abstract

This thesis develops mathematical programming frameworks to operate electric autonomous vehicles in the context of ride-sharing services. The introduced problem is a novel variant of the Dial-a-Ride Problem (DARP), denoted by the electric Autonomous Dial-a-Ride Problem (e-ADARP), and includes battery management and autonomous aspects along with classic dial-a-ride features. The e-ADARP operational frameworks undertaken in this thesis consider static problems, in which demand is assumed to be known in advance, and dynamic problems, in which demand is revealed on-line.

The *static* e-ADARP is formulated as a Mixed-Integer Linear Program and solved through a Branch-and-Cut framework, enhanced by problem-specific valid inequalities and lifted inequalities from the literature, as well as purpose-based separation heuristics. Computational experiments are performed on adapted benchmark instances from the DARP literature and on instances based on real data. Results show that the problem-specific valid inequalities are among the most effective and especially useful when battery management aspects are relevant.

The *dynamic* e-ADARP is developed through a simulation-based optimization approach, which incorporates a new extension to the family of large neighborhood search (LNS) metaheuristics. The extension considers a machine learning component which exploits historical information to select destroy-repair operators from a pool of competing algorithms, all along the search. Computational experiments are performed on dynamic e-ADARP instances based on real data. Results show that combining machine learning within LNS-based metaheuristics produces a competitive alternative to benchmark methodologies from the literature and in the context of on-line operations.

Finally, the e-ADARP is a hardly-constrained problem which is challenged by the addition of autonomous and battery management aspects within the vehicle scheduling algorithm. As such, this thesis proposes a novel scheduling procedure for the e-ADARP, which aims at finding minimal excess-time and battery-feasible schedules for fixed routes. Results on instances based on real data show that the algorithm is able to efficiently return optimal scheduling solutions.

Keywords: dial-a-ride problem, electric autonomous vehicles, static problem, dynamic problem, branch-and-cut, valid inequalities, large neighborhood search, machine learning, scheduling

Prefazione

La presente Tesi sviluppa modelli di programmazione matematica per il funzionamento di veicoli autonomi elettrici nel contesto di servizi ride-sharing. Il problema introdotto è una nuova variante del Dial-a-Ride Problem (DARP), indicato come l'electric Autonomous Dial-a-Ride Problem (e-ADARP), ed include aspetti di gestione della guida autonoma e delle batterie al contempo delle classiche funzionalità dial-a-ride. I quadri operativi previsti considerano problemi statici, nei quali la domanda è conosciuta in anticipo, e problemi dinamici, nei quali la domanda è rivelata on-line.

L'e-ADARP statico è formulato tramite un programma lineare a numeri interi ed è risolto tramite un algoritmo Branch-and-Cut, arricchito da tagli specifici ed estratti dalla letteratura, oltre ad efficaci procedure di separazione. Esperimenti computazionali sono eseguiti su istanze di riferimento adattate dalla letteratura DARP e su nuove instanze estratte da dati reali. I risultati mostrano che i tagli introdotti sono tra i più efficaci e particolarmente utili quando gli aspetti di gestione delle batterie sono rilevanti.

L'e-ADARP dinamico è sviluppato attraverso un approccio di ottimizzazione basato sulla simulazione, che incorpora una nuova estensione alla famiglia della metaeuristica Large Neighborhood Search (LNS). L'estensione considera una componente di machine learning che sfrutta informazioni storiche per selezionare operatori di distruzione-riparazione da un pool di algoritmi concorrenti, lungo tutta la ricerca. Gli esperimenti computazionali sono eseguiti su istanze dinamiche sulla base di dati reali. I risultati mostrano che l'unione del machine learning e di metauristici LNS produce un'alternativa competitiva alle metodologie proposte in letteratura e nel contesto di operazioni on-line.

Infine, l'e-ADARP è un problema duramente vincolato che è messo alla prova dall'aggiunta di aspetti di gestione dell' autonomia e delle batterie all'interno dell'algoritmo di schedulazione dei veicoli. Per questo motivo, la presente Tesi propone una nuova procedura di schedulazione per l'e-ADARP, che mira a minimizzare i tempi di viaggio degli utenti e al contempo pianificare i tempi di ricarica dei veicoli per percorsi fissi. Risultati su istanze basate su dati reali mostrano che l'algorithmo è in grado di restituire soluzioni di schedulazione ottimale in modo efficiente.

Parole chiave: dial-a-ride problem, veicoli a guida autonoma, problema statico, problema dinamico, branch-and-cut, tagli, large neighborhood search, machine learning, schedulazione

Contents

List of Figures **xv**

List of Tables **xvii**

List of Abbreviations **xix**

1 Problem statement **1**

1.1 Context and Motivation 1

1.2 State of the Art 3

1.2.1 The Dial-a-Ride Problem 3

1.2.2 Exact and Approximate Solution Methodologies 5

1.2.3 Local Search-Based Metaheuristics 7

1.2.4 Machine Learning for Operations Research 9

1.3 Contribution and Organization 9

2 A Branch-and-Cut Framework for the Static Electric Autonomous Dial-a-Ride Problem **13**

2.1 Introduction 13

2.2 Mathematical Formulations 14

2.2.1 Three-Index Formulation 14

2.2.2 Two-Index Formulation 18

2.3 Valid Inequalities 20

2.3.1 Valid Inequalities from the Vehicle Routing Literature 20

2.4 Valid Inequalities for the e-ADARP 21

2.4.1 Infeasible Path Inequalities – Battery Capacity 21

2.4.2 Infeasible Path Inequalities – Charging Stations “Walls" . . . 22

2.4.3 Charging Stations and Time Windows Constraints 23

2.4.4 Charging Visits Constraint 23

2.5 Branch-and-Cut Algorithm 24

2.5.1 Initialization 24

2.5.2 Separation Heuristics 26

2.6 Numerical Experiments 27

2.6.1 Test Instances 27

2.6.2 Branch-and-Cut Results 30
2.6.3 Sensitivity Analysis 34
2.7 Summary 38

3 A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic Electric Autonomous Dial-a-Ride Problem 41
3.1 Introduction 41
3.2 Problem Definition 43
3.2.1 Static Sub-Problems in the Dynamic e-ADARP 45
3.3 Solution Approach 47
3.3.1 Event-Based Simulation 48
3.3.2 Two-phase Metaheuristic for Static e-ADARP Sub-Problems 49
3.3.3 Second Phase: Machine Learning-Based Large Neighborhood Search 51
3.4 Prediction Problem 53
3.4.1 Definition 54
3.4.2 Machine Learning Approximation 54
3.5 Dataset Labeling 55
3.6 Feature Selection 57
3.7 Numerical Experiments 60
3.7.1 Labeled dataset 61
3.7.2 Machine Learning Results 64
3.7.3 Feature Importance 66
3.7.4 ML-LNS Results 67
3.8 Summary 75

4 Scheduling Algorithm and Battery Management Heuristic for the Electric Autonomous Dial-a-Ride Problem 77
4.1 Introduction 77
4.2 Scheduling Problem 79
4.3 Scheduling Procedure 81
4.4 Battery Management Heuristic 84
4.5 Numerical Experiments 85
4.6 Summary 87

5 Conclusion 89
5.1 Main Findings 89
5.2 Practical Implications 90
5.3 Future Research Directions 91

References 93

Curriculum vitae **103**

List of Figures

2.1 Inequalities (2.56) for the case in which $\mathbb{U} = \{i, j\}$, $\mathbb{S} = \{k, g, n+k\}$, $\mathbb{T} = \{n+i, n+j\}$, and $\mathcal{S} = \{s_1, s_2, s_3\}$ 21
2.2 Inequalities (2.58) for the case in which $\mathbb{P} = \{i, m_1, m_2, \ldots, m_q, s_2\}$ and $|\mathcal{S}| = 3$ 23
2.3 Inequalities (2.59) for the case in which $\mathbb{P}'' = \{s_2, m_1, m_2, \ldots, m_1, n+i\}$ and $|\mathcal{S}| = 3$ 23
2.4 Inequalities (2.60): Charging stations and time windows constraints 24
2.5 Instance a4-16: (a) Pickup and dropoff locations; (b) Pickups and dropoffs time windows after tightening. Note that the pickup location of any user with pickup ID i is denoted by i+16 29
2.6 Instance u4-16: (a) Pickup and dropoff locations; (b) Pickups and dropoffs time windows after tightening. Note that the pickup location of any user with pickup ID i is denoted by i+16 29

3.1 Flowchart for the event-based simulation 48
3.2 Considered re-optimization process 53
3.3 Example of the algorithm performance distributions after M trials . 57
3.4 Label distributions: (a) Expected objective function improvement, (b) Accepted neighborhood moves by SA, (c) Scaled expected objective function improvement. 63
3.5 Labeled dataset and simple insertion: (a) Difference in terms of number of accepted requests, (b) Average CPU time at each iteration of the search 64
3.6 Distribution of the absolute errors between observed and predicted estimates in the training and testing dataset 66
3.7 Average feature importance 67
3.8 Distributions on the total number of accepted requests per algorithm 69
3.9 Distributions on the average objective function improvement per LNS search 70
3.10 Distributions on the number of re-optimized sub-problems per algorithm 71

3.11 Distributions on the percentage of times algorithms introduce the upcoming request during the search 73
3.12 Distributions on average number of LNS iterations per algorithm . . 74

List of Tables

1.1 Thesis structure, content, and contribution 12

2.1 Problem sets, parameters, and decision variables 15
2.2 3-indexed formulation for the e-ADARP 16
2.3 2-indexed formulation for the e-ADARP 19
2.4 Benchmark Instances: (a) Instances adapted from Cordeau (2006); (b) Instances from Uber ride-shares in San Francisco, CA (USA). . 28
2.5 Instances adapted from Cordeau (2006): Comparison between e-ADARP3 and e-ADARP2 31
2.6 Instances from Uber: Comparison between the e-ADARP3 model and the e-ADARP2 model 32
2.7 Instances from Uber: Root node lower bounds as a percentage of the upper bound 33
2.8 Instances from Uber: Solution quality and performance when increasing the maximum number of charging visits per station. 35
2.9 Instances from Uber, Electric fleet compared to an internal combustion (IC) fleet: Number of utilized vehicles (Vehs), Total vehicle travel time cost (TT), Users excess ride time cost (ERT), Total number of visited charging stations en-route (n_{S_R}), Total charging time en-route (E_{S_R}), Total number of visited charging stations off-route (n_{S_D}), Total charging time off-route (E_{S_D}) 36
2.10 Instances from Cordeau (2006): Comparison between the e-ADARP2 model, the e-ADARP2 model when employing an internal combustion fleet, the e-ADARP2 model when only considering total vehicle travel time in the objective function, the standard DARP results. . 37

3.1 3-indexed formulation for static e-ADARP sub-problems at time h . 46
3.2 Features description 58
3.3 Performance measures obtained for the 9 RF regressors on the train and test dataset: Mean absolute error (MAE), root mean square error (RMSE), coefficient of determination (R^2), out of bag error (OOB). 65

4.1 Comparison between the scheduling solution of the LP and the route evaluation procedure. In order of appearance: instance name, total number of visited stations, total vehicle recharging time from the LP solution [min], total vehicle waiting time from the LP solution [min], objective function value (total user excess ride time) from the LP solution [min], total cpu time to retrieve the LP solution [sec]; analogous results for the route evaluation procedure. 86

List of Abbreviations

ALNS	Adaptive large neighborhood search
B&C	Branch and Cut
CART	Classification And Regression Trees
DARP	Dial-a-ride problem
e-ADARP	Electric autonomous dial-a-ride problem
e-AVs	Electric autonomous vehicles
LNS	Large neighborhood search
LP	Linear program
LS	Local Search
MILP	Mixed integer linear program
ML	Machine learning
ML-LNS	Machine learning-based large neighborhood search
PDP	Pickup and delivery problem
RF	Random Forests
SOC	State of charge
TS	Tabu Search
TSP	Traveling salesman problem
VRP	Vehicle routing problem

1

Problem statement

This chapter is based on the articles:

- C. Bongiovanni, M. Kaspi, and N. Geroliminis (2019). “The Electric Autonomous Dial-a-Ride Problem”. In: *Transportation Research Part B: Methodological* 122, pp. 436–456
- C. Bongiovanni, M. Kaspi, J.-F. Cordeau, and N. Geroliminis (2020). “A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic Electric Autonomous Dial-a-Ride Problem”. Working paper
- C. Bongiovanni, M. Kaspi, and N. Geroliminis (2020). “Scheduling Algorithm and Battery Management Heuristic for the electric Autonomous Dial-a-Ride Problem”. Working paper

This chapter introduces the problem at the center of this thesis (Section 1.1), provides a review of the relevant literature (Section 1.2), and outlines the thesis objective, contribution, and structure (Section 1.3).

1.1 Context and Motivation

Ride-sharing has modified urban mobility by offering reliable and affordable door-to-door services at any time. This relatively new business model is of great interest in the area of city logistics and passenger mobility, as it creates a convenient alternative to private cars and public transport. In the last decade, ride-sharing has attracted a non-negligible share of the demand (Soper, 2015), making up to 50% of rides in San Francisco after one year from its introduction. Given the constant increase in user requests, ride-sharing businesses are currently planning to expand their portfolio to include the use of electric autonomous vehicles (e-AVs) (Perry, 2020).

The concept of carpooling is not new. However, up until today, it has been referred to as dial-a-ride transit (Dial, 1995) and developed at local scales, mostly providing service to the elders, the disabled, and within airports (Cervero, 1997). Such paratransit system offers scheduling and routing flexibility, similarly to ride-sharing, while maintaining characteristics of public transport and taxi services (e.g., many-to-many distribution, low accessibility costs, higher capacities, possibility to book in advance). The rise in ride-sharing demand currently calls for the re-design of dial-a-ride transit, which needs to be efficiently modeled to face high demand pressures at large scale.

The introduction of e-AVs is expected to have positive effects on urban mobility, public transport safety, and the environment. With respect to urban mobility, according to a recent study, ride-sharing could reduce taxi traffic by 75 percent while still serving the same demand (Alonso-Mora et al., 2017). Furthermore, Talebpour and Mahmassani (2016) indicate that urban congestion could be highly decreased by the introduction of e-AVs, improving traffic flow stability. Although road safety implications are still under debate, Fagnant and Kockelman (2015) demonstrate that e-AVs may positively contribute to reducing crashes, with the major cause of collision being driver-related. Finally, since fully-autonomous vehicles are electric, an increased penetration of e-AVs within ride-sharing transit may result in a significant reduction of local greenhouse gas emissions as shown in Greenblatt and Saxena (2015). Indeed, a substantial drop in fuel-ignition transport, from cars to delivery trucks, is key to reducing urban pollution, as confirmed by the recent world-wide covid-19 lockdown (Barbiroglio, 2020). As such, public agencies in multiple cities, such as Milan among others, are currently planning to devote urban street space to more sustainable means of transport (Laker, 2020).

The use of e-AVs is expected to enhance ride-sharing operations by offering several new opportunities. First, e-AVs provide more flexibility to efficiently modify vehicle plans according to changing conditions. Such changes may not only correspond to the arrival of new transportation requests but also to other unexpected events, such as an increase in traffic congestion, although this last aspect is not considered in this thesis. That is, differently from human-driven vehicles, in which the number of deviations must be controlled (Ferrucci and Bock, 2015), the dispatching system can easily divert e-AVs as often as desired in the course of operations. Second, e-AVs can operate non-stop, being driverless and undependent from driver shifts. This feature may help saving vehicle deadhead miles to decentralized depots and provide higher service levels. Third, the type of service provided by e-AVs does not need to be pre-defined (e.g. taxi service, ride-sharing service, public transport) and can instead be automatically adapted depending on new demand.

Despite the numerous benefits, the introduction of electric autonomous ride-sharing into urban mobility poses a number of challenges related to its design at the tactical and strategical level. For example, the practical use of e-AVs for ride-sharing assumes that the fleet has been appropriately sized with respect to the demand and that electric chargers are strategically and sufficiently deployed at the urban level (e.g. Tran et al., 2018; Diana, Dessouky, and Xia, 2006). The work undertaken in this thesis prescinds from tactical and strategic challenges, which can be taken off-line, and instead focuses on decisions at the operational level, which can be tackled off-line (i.e. if trips are booked in-advance) or on-line (i.e. if trips are revealed in real-time). In both cases, the planning process needs to consider the optimization of vehicle battery levels, decisions regarding detours to charging stations, recharging times, and destination depots together with the classic dial-a-ride features. The combination of all such features undoubtedly pose additional operational challenges, most especially for real-time operations as decisions need to be taken within a very limited computing time. As such, new optimization frameworks are needed to treat the added complexity induced by the use of e-AVs for urban ride-sharing. For a complete review of operational benefits and challenges on the use of autonomous vehicles for ride-sharing, the reader is referred to Hyland and Mahmassani (2020).

This thesis develops novel optimization frameworks for electric autonomous ride-sharing. It starts by providing an exact optimization approach to be employed in the context of off-line trip reservations and continues with the development of a machine learning-based metaheuristic which is employed in the context of on-line trip reservations. The thesis concludes by proposing a novel scheduling algorithm that is used to efficiently tackle battery aspects of the e-ADARP while maximizing the provided level of service.

1.2 State of the Art

1.2.1 The Dial-a-Ride Problem

The dial-a-ride problem (DARP) is a class of combinatorial optimization problems which arises in the context of on-demand transportation systems. In its standard version, the problem consists of defining minimum cost routes and schedules for a fleet of conventional vehicles exiting a common depot and serving a set of customers with given pickup and dropoff locations, as well as corresponding pickup or dropoff times. After serving all requests, and by the end of the driver shifts, the vehicles are required to return to a common destination depot. Typical operational limitations include capacity, duration, time-window, and ride-time constraints (Cordeau and Laporte, 2003).

The DARP is a generalization of well-known optimization problems, such as the traveling salesman problem (TSP) and the vehicle routing problem (VRP). It can also be viewed as a special case of the pickup and delivery problem (PDP) (Toth and Vigo, 2014). As noted in Cordeau and Laporte (2003), the DARP is more challenging than other routing problems due to the need to weight transportation cost and user inconvenience against each other. Indeed, a single operational objective does not provide an incentive to optimize service quality, although this last aspect may be controlled by imposing service-level constraints. As such, several DARP variants have considered a combination of operational and quality-related objectives (see section 3.3.2 in Molenbruch, Braekers, and Caris, 2017). In fact, considering a fixed feasible routing solution, the quality of service may be improved by minimizing a quality-oriented objective at no additional operational cost (Parragh, 2011). In the works in Parragh (2011) and Molenbruch et al. (2017), trasportation cost, measured by the total vehicle travel time, and user inconvenience, measured by the total user excess ride time, are both employed through a Pareto approach. Differently from other quality measures (e.g. total waiting time of all vehicles with passengers aboard), the consideration of user excess ride time allows to directly quantify user-specific costs.

The DARP literature can be divided into two main streams, namely static and dynamic DARP. In the first case, demand is fully known in advance and needs to be served, whereas in the second case demand is revealed on-line. No information about future requests is typically assumed, although some stochastic information (e.g. Albareda-Sambola, Fernández, and Laporte, 2014; Ichoua, Gendreau, and Potvin, 2006) and forecasts may be used (e.g. Peleda et al., 2019; Ferrucci and Bock, 2016; Ferrucci, Bock, and Gendreau, 2013). Given the inherent uncertainty about demand, transportation requests are allowed to be denied in the dynamic DARP. Assuming that each denied request results into a homogeneous profit loss, solution quality is primarily measured by the total number of served requests and secondly by operational cost (e.g. Berbeglia, Cordeau, and Laporte, 2012; Attanasio et al., 2004).

Battery-management aspects for electric vehicles have been widely studied in several vehicle routing problems. For example, the electric and green VRP (Erdoğan and Miller-Hooks, 2012; Schneider, Stenger, and Goeke, 2014; Felipe et al., 2014; Goeke and Schneider, 2015; Desaulniers et al., 2016; Pelletier, Jabali, and Laporte, 2016; Keskin and Çatay, 2016; Schiffer and Walther, 2017), the hybrid electric TSP (Arslan, Yıldız, and Karaşan, 2015; Doppstadt, Koberstein, and Vigo, 2016), and the mix vehicle routing problem with time windows and recharge stations (Baldacci, Battarra, and Vigo, 2009; Hiermann et al., 2016). Electric VRP studies frequently assume that discharge times are linearly dependent on travel times (e.g. Schneider, Stenger, and Goeke, 2014; Desaulniers et al., 2016) or derived from energy consumption models (e.g. Goeke and Schneider, 2015;

Genikomasakis and Mitrentsis, 2017; Pelletier, Jabali, and Laporte, 2018). Typical recharge policies from the literature include (1) full recharge (e.g. Schneider, Stenger, and Goeke, 2014) (2) battery swapping (e.g. Masmoudi et al., 2018) (3) partial recharge while neglecting time windows (Felipe et al., 2014), and (4) partial recharge without neglecting time-windows (Desaulniers et al., 2016). Note that all of the fore-mentioned policies do not impose any minimal battery level for vehicles to return to their destination depots.

Finally, although several DARP problem variants have been proposed in the literature, a unified model combining battery management and vehicle autonomy, along with dial-a-ride features, remains unexplored. This is the scope undertaken in this thesis, which introduces the electric autonomous dial-a-ride problem (e-ADARP) and related solution methodologies for static and dynamic operations.

1.2.2 Exact and Approximate Solution Methodologies

The standard DARP, as well as its variants, can be modeled through Mixed Integer Linear Programming (MILP). Being a generalization of the TSP, the problem is NP-hard and needs the development of appropriate solution methodologies. Specifically, there are two possible classes of solution techniques for the DARP and those are exact and approximate solution methodologies. Exact solution methodologies are designed to find the global optimum and attempt to decrease computational complexity by supplementing the branch-and-bound algorithm (Balas and Toth, 1983) with problem-dependent considerations controlling the depth, the chosen variables, and the branching of the tree. Approximate solution methodologies are instead designed to fastly return a local optimum and are typically composed of problem-dependent heuristics.

Solution methodologies for static and dynamic DARP problems share some similarities in that both problems may be theoretically solved by employing exact and approximate solution techniques. However, dynamic versions of the problem are highly constrained by computational time and affected by uncertainty, given that information is stochastic and revealed over time. As such, exact solution approaches are typically only applied to static versions of the DARP, while both problems can be solved through approximate solution techniques. Exact solution approaches for the static DARP and related routing problems typically include: branch-and-cut algorithms – B&C (e.g. Lu and Dessouky, 2004; Cordeau, 2006; Ropke, Cordeau, and Laporte, 2007; Parragh, 2011; Braekers, Caris, and Janssens, 2014; Braekers and Kovacs, 2016) and branch-and-cut-and-price algorithms (Ropke and Cordeau, 2009; Baldacci, Bartolini, and Mingozzi, 2011; Parragh and Schmid, 2013; Gschwind and Inrich, 2014). Approximate solution approaches for both the static and dynamic DARP and related problem include: tabu search – TS (Cordeau, Laporte, and Mercier, 2001; Cordeau and Laporte, 2003; Attanasio et al., 2004; Berbeglia, Cordeau, and Laporte, 2012), local search – LS (Savelsbergh, 1985;

Healy and Moll, 1995; Funke, Grünert, and Irnich, 2005; Molenbruch et al., 2017), and large neighborhood search – LNS (Shaw, 1997; Ropke and Pisinger, 2006; Parragh, Doerner, and Hartl, 2010; Li et al., 2016; Gschwind and Drexl, 2019; Sacramento, Pisinger, and Ropke, 2019). For the dynamic DARP, approximate solution methodologies are typically employed in a two-phase apprach (e.g. Archetti, Fernández, and Huerta-Muñoz, 2018; Schilde, Doerner, and Hartl, 2011b; Parragh et al., 2009). The first phase attempts to feasibly assign requests to vehicles through a polynomial-time insertion heuristic (Jaw et al., 1986; Potvin and Rousseau, 1993; Diana and Dessouky, 2004; Coslovich, Pesenti, and Ukovich, 2006; Parragh et al., 2009; Marković et al., 2015). The second phase is designed to re-optimize previous insertion decisions through one of the previously mentioned approximate solution methodologies but most typically through local search-based metaheuristics. For a comprehensive review on DARP problem variants and objectives and related solution methods, the reader is referred to Molenbruch, Braekers, and Caris (2017).

The size of the problems which can be solved through exact and approximate solution methodologies are very different. Exact methods can only solve problems of limited size (Baugh, Kakivaya, and Stone, 1998), with the largest problems currently involving 8 vehicles and 96 customers (Ropke, Cordeau, and Laporte, 2007). Approximate solution methods can find a local optimum to problems of very large size, going up to hundreds of vehicles and thousands of customers (e.g. Gschwind and Drexl, 2019; Ropke and Pisinger, 2006; Molenbruch et al., 2017). Both approaches are computationally affected by each added feature to the standard version of the DARP. For example, generalized versions of the static DARP have been addressed by Braekers, Caris, and Janssens (2014) and Parragh (2011). Braekers, Caris, and Janssens (2014) considered heterogeneous vehicles, multiple depots, multiple user types and could solve instances of up to 8 vehicles and 80 customers. Parragh (2011) further considered a weighted-sum objective function consisting of the total routing cost and wait time for all vehicles with passengers aboard. The objective function included a time-dependent criterion, which in turn did not allow to reduce the problem size by avoiding timing decision variables and constraints. The problem resulted in a highly-constrained model, which could only be solved for instances of up to 4 vehicles and 40 customers.

In order to limit the computational burden from the DARP, much research has been focused on the development of scheduling heuristics to ease the solution process. Given fixed routing sequences, the DARP can be reduced to a scheduling problem, which can be formulated as a linear program (LP) aiming at finding the right service start times for the nodes composing the given sequences. Some research has been specifically focused on testing feasibility of given routes, without necessarily returning optimized schedules (e.g. Hunsaker and Savelsbergh, 2002; Berbeglia, Pesant, and Rousseau, 2011; Häme and Hakula, 2015). Other works have instead focused on providing feasible routes, which are heuristically optimized (e.g. Cordeau and Laporte, 2003; Parragh et al., 2009; Molenbruch et al., 2017). Specifically in the works of Parragh et al. (2009) and Molenbruch et al. (2017), the scheduling heuristics are excess-time oriented. However, in the case of the e-ADARP excess-time oriented schedules need to further consider battery management aspects. Specifically, they need to find battery feasible routes corresponding to the maximal level of service.

The e-ADARP is a new challenging generalization of the DARP which is modeled and solved through exact and approximate solution methodologies. Added challenges include battery management, optional charging facilities, recharging plans, and destination depots, along with classic dial-a-ride features. This thesis develops a branch-and-cut approach for the static e-ADARP and a two-phase metaheuristic for the dynamic e-ADARP. The two-phase metaheuristic employs a novel e-ADARP scheduling algorithm which returns battery-feasible schedules while minimizing user ride time. In particular, the second phase of the two-phase approach is composed of a novel local search-based metaheuristic employing a machine learning module to efficiently drive the search towards good areas of the solution space. The following sections focus on the relevant literature on local search-based metaheuristics and provide a short review of the literature combining machine learning within optimizataion algorithms.

1.2.3 Local Search-Based Metaheuristics

Local search or local search-based metaheuristics are popular optimization approaches for vehicle routing problems (Funke, Grünert, and Irnich, 2005). These methods iteratively transform a given solution through elementary neighborhood moves in the direction of the objective function (e.g. Bertsimas, Jaillet, and Martin, 2019; Simonetto, Monteil, and Gambella, 2019; Alonso-Mora et al., 2017). Each elementary move is composed of a simple heuristic shuffling a limited number of requests between vehicles, e.g. 2-opt. Large Neigborhood Search extends the local search paradigm by considering moves shuffling a larger share of requests between vehicles. Each move may be performed through a more complex neighborhood structure, which sequentially destroys and repairs a given solution. The destruction degree, i.e. the total number of requests to be shuffled, is typically drawn from a uniform distribution supported on a bound interval, i.e. between a minimum and a maximum destruction level. If the bound is too restrictive, the incumbent solution may be modified only marginally. As a result, the search may have difficulties in moving towards promising neighborhoods and get trapped in a local minimum. One way of dealing with this drawback is to loosen the interval bound, allowing the algorithm to re-arrange a higher percentage of the vehicle requests. Nevertheless, higher destruction degrees typically have a negative effect on computational time (Pisinger and Ropke, 2010). Another methodology which is frequently employed to try and escape local minima explores non-improving and infeasible solutions (e.g.

Cordeau, Laporte, and Mercier, 2001; Cordeau and Laporte, 2003). Worsening solutions may be generally explored by employing an acceptance criterion based on simulated annealing – SA (Nikolaev and Jacobson, 2010). Simulated annealing employs an analogy with metallurgy by using a probability function which depends on an initial temperature and a cooling rate controlling the likelihood of accepting deteriorating solutions over successive iterations. If worsening solutions are accepted, the solution is expected to iteratively recover towards promising areas of the search space. Such recovery cannot be guaranteed as it highly depends on the type of operator and destruction degree being employed. The recovery is finally bound by a maximal number of successive non-improving steps, which may determine an early exit from the search. Originally introduced by Shaw (1997), LNS has been widely applied to solve static large-scale problems (e.g. Molenbruch et al., 2017; Masmoudi et al., 2016; Parragh, Doerner, and Hartl, 2010) and dynamic problems (e.g. Schilde, Doerner, and Hartl, 2014; Schilde, Doerner, and Hartl, 2011a; Attanasio et al., 2004). For a review of LNS, the reader is referred to section 13.2 in Pisinger and Ropke (2010).

Recent literature has demonstrated added value in switching between multiple destroy-repair couples during the search by adopting a metaheuristic approach (Elshaer and Awad, 2020). Metaheuristics automate the choice of the destroy-repair methods to be employed at any iteration in the search by proposing a selection mechanism. Metaheuristics are also frequently noted as hyper-heuristics (e.g. Burke et al., 2013), although the second typically refers to a combination of metaheuristics rather than destroy-repair methods. In the vehicle routing literature, multiple recent studies have employed a selection mechanism by considering a score function providing a measure for the success of selected destroy-repair operators on previous iterations (e.g. Sacramento, Pisinger, and Ropke, 2019; Gschwind and Drexl, 2019; Li et al., 2016; Goeke and Schneider, 2015). This selection mechanism, introduced in Ropke and Pisinger (2006), is denoted by adaptive large neighborhood search (ALNS). For a review of ALNS, the reader is referred to section 13.2.1 in Pisinger and Ropke (2010). In particular, the score of each destroy-repair couple is initialized according to a roulette wheel mechanism and it is updated through a linear function which employs statistics on the solution quality from accepted moves during previous iterations. The update magnitude is scaled by a reaction factor r, controlling how quickly the linear function adapts to changes in the statistics. A warm-up time of hundreds of iterations is typically needed to tune the several destroy-repair scores before being able to make statistically-informed choices between competing operators. Indeed, ALNS is typically employed for large static problems, which are not as tightly bound by computational time as for dynamic settings. Chapter 3 from this thesis proposes an alternative approach that allows to eliminate the warm-up time required to tune the adaptive mechanism and that is more suitable for real-time settings.

1.2.4 Machine Learning for Operations Research

Recent advance in artificial intelligence has fostered new research directions at the intersection between machine learning (ML) and optimization. While optimization methods have been often employed in ML algorithms (e.g. Bertsimas and Dunn, 2017; Gunluk et al., 2016; Bertsimas and Shioda, 2007), the use of ML in optimization algorithms is relatively new and has been focusing on the following tasks: (1) ML as an approximation tool to tackle time-consuming tasks in discrete optimization algorithms (Bonami, Lodi, and Zarpellon, 2018; Kruber, Lübbecke, and Parmentier, 2017; Lodi and Zarpellon, 2017), (2) ML to heuristically solve discrete optimization problems (Larsen et al., 2019; Bengio, Lodi, and Prouvost, 2018;Vinyals, Fortunato, and Jaitly, 2015) (3) ML to choose among a number of competing algorithms to solve hard optimization problems (Kerschke et al., 2019; Malitsky et al., 2013; Gomes and Selman, 2001). Specifically, the last task is also known as meta-learning for algorithm porfolios and has focused on automating the selection of solution frameworks by extracting significant problem features for a number of discrete combinatorial problems, including the TSP. Finally, increasing interest has been shown in the use of machine learning within metaheuristic approaches (e.g. Vigo et al., 2019). However, this research area is new and has not yet been explored. The idea, developed in Chapter 3, is to exploit past information, i.e. from previously solved similar problems, to quickly guide the search towards good solutions, i.e. avoiding a warm-up period. Other than driving the search, the use of machine learning within metaheuristics allows to identify features of the dial-a-ride problem (and more generally VRP) that mostly impact the solution quality and to classify new problem instances accordingly.

1.3 Contribution and Organization

This thesis extends the standard static and dynamic DARP by considering the use of electric autonomous vehicles. The extended problem is referred to as the electric autonomous dial-a-ride problem (e-ADARP). In its static version, the e-ADARP aims at minimizing a weighted objective function consisting of the total travel time of all vehicles and excess ride-time of all users. In its dynamic version, the e-ADARP additionally aims at maximizing the number of accepted requests. Different weights may be chosen by practicioners by analyzing the trade-off between the terms in the objective function on each specific case study. Note that another approach may employ a (hirerchical) bi-objective approach instead of a weighted objective function. However, in this case, practicioners would have to analyze the impact of each component of the objective function through a Pareto approach.

As in the work of Schneider, Stenger, and Goeke (2014) and Desaulniers et al. (2016), in this thesis battery consumption is assumed to be constant and independent from the load, speed, and state-of-charge (SOC) when vehicles travel between nodes. This assumption is suitable for cases where vehicle load is negligible with respect to the curb weight of the vehicle and velocity is constant or experiences infrequent variations (Desaulniers et al., 2016; Pelletier et al., 2017a). This is the case for the electric autonomous vehicles considered in this thesis, which are characterized by limited capacity and cruise speed. Furthermore, it is assumed that vehicles can visit multiple charging stations along a route, can partially recharge, and need to return with minimal battery levels at the destination depots. With respect to recharge phases, it is assumed that the SOC linearly increases with time by means of a recharge rate. This rate can be inferred from the model presented in Pelletier et al. (2017a) and applied in Pelletier, Jabali, and Laporte (2018).

While some of the features of the e-ADARP relate to the electric nature of the fleet, some are derived from considerations associated with the vehicle autonomy. In particular, autonomous vehicles can operate on a non-stop schedule and so the provided service is not limited by the driver shifts. As a result, the e-ADARP does not enforce vehicle maximum route-duration constraints, introducing savings in dead-head miles and allowing for higher service levels. Additionally, providing a non-stop service eliminates the need to pre-define common depots from which drivers start and end their shifts. Instead, the vehicles can continuosly exit and re-enter multiple depots within the service zone, while waiting for new instructions. The use of multiple origin and destination depots are nowadays commonly adopted in extensions of the standard DARP (e.g. Parragh, 2011; Braekers, Caris, and Janssens, 2014). However, the assignment of vehicles to depots is normally predetermined, instead of being selected in the decision process. Consequently, in the e-ADARP the planning and battery-management problem is further supplemented by a vehicle-to-depot assignment problem which may additionally consider predictions upon future demand to strategically relocate vehicles during the day. Additional autonomy features are most evident in the context of dynamic ride-sharing operations. In particular, the operation of autonomous vehicles offers more flexibility to efficiently modify vehicle plans according to changing or unexpected events. Such events may not only correspond to the arrival of new transportation requests but also to unexpected increase in traffic congestion, modified availabilities at charging facilities, and vehicle breakdowns. That is, differently from human-driven vehicles, in which the number of deviations must be controlled (Ferrucci and Bock, 2015), the dispatching system can easily divert e-AVs as often as desired in the course of operations. Consequently, the planning process needs to continuously re-optimize decisions regarding vehicle-trip assignments, detours to charging stations, recharging times, and destination depots together with the classic dial-a-ride features.

To summarize, compared to classical DARP, the e-ADARP is supplemented with the following features:

1. Battery-management;
2. Intermediate stops for vehicle recharge;
3. Heterogeneous vehicles in terms of capacity and initial battery inventories;
4. Vehicles initially located at different depots;
5. Vehicles return to a set of optional depots;
6. No restrictions on maximum route-durations are applied;
7. Vehicles continuously re-routed in the course of operations;
8. Weighted objective function considering operational and level of service costs.

Note that, most of the mentioned features can be independently found in the vehicle routing literature. However, their combination has not yet been explored and results in a new highly-constrained problem variant to be studied.

The scope of this thesis is to model and solve the static and dynamic e-ADARP through a novel exact and metaheuristic approach. The *static* e-ADARP is formulated as a 3-indexed or a 2-indexed Mixed-Integer Linear Problem (MILP) which is solved through a branch-and-cut algorithm. The solution approach is enhanced by new problem-specific valid inequalities which are separated through ad-hoc procedures. The numerical experiments are obtained by supplementing benchmark instances from the literature (Cordeau, 2006) with charging stations and battery specifications, and by extracting new benchmark instances derived from real data from Uber Technologies Inc. in San Francisco. The *dynamic* e-ADARP is solved through a two-phase heuristic approach within an event-based simulation framework. The second phase is composed of a machine learning-based large neighborhood search (ML-LNS). The machine learning component is employed to iteratively direct the search towards promising areas of the search space. The proposed metaheuristic is compared to ALNS (Ropke and Pisinger, 2006) on real dynamic instances from Uber Technologies Inc. in San Francisco. The scheduling algorithm employed to plan and evaluate instances of the dynamic e-ADARP is solved through an e-ADARP-specific *scheduling procedure*. Namely, the procedure combines an exact scheduling algorithm minimizing user excess ride time and a battery management heuristic. Scheduling solutions from the proposed procedure are compared to those obtained from a linear program on the vehicle routes derived for the static e-ADARP. Finally, the rest of this thesis is structured into five chapters, whose main objectives and contributions are summarized in Table 1.1.

Table 1.1: Thesis structure, content, and contribution

Chapter	Title	Contribution
Ch. 2	A Branch-and-Cut framework for the static e-ADARP (from C. Bongiovanni, M. Kaspi, and N. Geroliminis, 2019)	3-indexed and 2-indexed MILP B&C algorithm Problem-dependent valid inequalities Lifted valid inequalities from literature Separation heuristics Static instances from the literature Static instances from real data
Ch. 3	A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic electric Autonomous Dial-a-Ride Problem (from C. Bongiovanni, M. Kaspi, J.-F. Cordeau, and N. Geroliminis, 2020)	Event-based simulation framework Two-phase heuristic approach First phase: Greedy insertion heuristic Second phase: Machine learning-based LNS Labeled dataset and feature extraction Dynamic instances from real data
Ch. 4	A Scheduling Algorithm and Battery Management Heuristic for the electric Autonomous Dial-a-Ride Problem (from C. Bongiovanni, M. Kaspi, and N. Geroliminis, 2020)	e-ADARP scheduling problem Excess-time optimal scheduling procedure Battery management heuristic Static instances from real data
Ch. 5	Conclusion and future research directions	

2

A Branch-and-Cut Framework for the Static Electric Autonomous Dial-a-Ride Problem

This chapter is based on the article:

- C. Bongiovanni, M. Kaspi, and N. Geroliminis (2019). “The Electric Autonomous Dial-a-Ride Problem”. In: *Transportation Research Part B: Methodological* 122, pp. 436–456

2.1 Introduction

This chapter models and solves the static electric autonomous dial-a-ride problem, whose main differences with respect to the standard DARP are introduced in Chapter 1.3. There are three primary contributions to this chapter. First, we introduce the e-ADARP and formulate it as a MILP. Second, we devise a B&C algorithm and propose new problem-specific valid inequalities. Third, we supplement benchmark instances from literature (Cordeau, 2006) with charging stations and battery specifications, and provide new benchmark instances derived from real data from Uber Technologies Inc. in San Francisco. The rest of the chapter is organized as follows: in Section 2.2 we introduce a 3-index and a 2-index model for the e-ADARP; Section 2.3 and Section 2.4 define several valid inequalities, including new inequalities specifically designed for the e-ADARP; a branch-and-cut framework that utilizes these valid inequalities is then introduced in Section 2.5; computational experiments are reported in Section 2.6, followed by a summary and future research directions in Section 2.7.

2.2 Mathematical Formulations

2.2.1 Three-Index Formulation

Consider a complete directed graph $\mathbb{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ denotes the set of vertices and $\mathcal{A}$ the set of edges. Also consider a set $\mathcal{K} = \{1, \ldots, k\}$ of electric autonomous vehicles stationed at origin depots $\{o_1, \ldots, o_k\} \in \mathcal{O}$. The vehicles are heterogeneous in terms of their capacities C^k (with total maximum vehicle capacity $C = \max_{k \in \mathcal{K}} C^k$), homogeneous in terms of their battery capacities Q, and might feature different initial battery inventories B_o^k. The vehicles are utilized to give service to n users, specifying their pickup locations $\mathcal{P} = \{1, ..., n\}$, dropoff locations $\mathcal{D} = \{n + 1..., 2n\}$, and time windows $[arr_i,\ dep_i]$ around their desired arrival times. All user requests are known at the start of the planning period and need to be served. Furthermore, maximum user ride times u_i are imposed to limit the time users spend on-board the vehicles. Other than the origin depots and pickup and dropoff locations, the set of vertices $\mathcal{V}$ also include charging stations $\mathcal{S}$ and optional destination depots $\mathcal{F}$. We consider that each vehicle can return to one optional destination depot in $\mathcal{F}$. Differently from the set of the origin depots $\mathcal{O}$, the cardinality of $\mathcal{F}$ might be higher than the number of vehicles and includes the nodes in $\mathcal{O}$.

Each location $i \in \mathcal{V}$ is characterized by the change in load l_i, which is positive at pickup locations, negative at dropoff locations, and is zero at all other locations. In addition, non-null service times d_i are defined at pickup and dropoff locations, which represent the time vehicles need in order to board users. The amount of energy recharged at the charging stations in $\mathcal{S}$ is proportional to the time spent at the facilities. Each charging facility $s \in \mathcal{S}$ can only be accessed by empty vehicles and is characterized by a recharge rate α_s. This rate indicates the amount of energy transferred per unit time (i.e. fast charging, slow charging). It is assumed that vehicles can partially recharge at the visited charging stations. Finally, the vehicle battery levels cannot be lower than a certain minimum state of charge (SOC) upon arrival at one of the optional destination depots $f \in \mathcal{F}$ by the end of the planning horizon T_p. Such battery levels are controlled by the minimum battery level ratio r.

The edges in $\mathcal{A}$ represent travel times $t_{i,j}$ between any two locations $i \in \mathcal{V}$ and $j \in \mathcal{V}$. The battery consumptions $\beta_{i,j}$ can be inferred from the travel times $t_{i,j} \in \mathcal{A}$ combined with other features by means of an energy consumption model (Goeke and Schneider, 2015; Pelletier et al., 2017a). We assume that the triangular inequality holds both for travel times and battery consumptions.

Table 2.1: Problem sets, parameters, and decision variables

Sets	
$\mathcal{P} = \{1, ..., n\}$	Set of pickup locations
$\mathcal{D} = \{n+1, ..., 2n\}$	Set of dropoff locations
$\mathcal{N} = \mathcal{P} \cup \mathcal{D}$	Set of pickup and dropoff locations
$\mathcal{K} = \{1, ..., k\}$	Set of available vehicles
$\mathcal{O}$	Set of origin depots for vehicles $k \in \mathcal{K}$, the origin of vehicle k is denoted by o^k
$\mathcal{F}$	Set of all available destination depots
$\mathcal{S}$	Set of all charging stations
$\mathcal{V} = \mathcal{N} \cup \mathcal{O} \cup \mathcal{F} \cup \mathcal{S}$	Set of all possible locations
Parameters	
$t_{i,j}$	Travel time from location $i \in V$ to location $j \in \mathcal{V}$
arr_i	Earliest time at which service can begin at $i \in \mathcal{V}$
dep_i	Latest time at which service can begin at $i \in \mathcal{V}$
d_i	Service duration at location $i \in \mathcal{V}$
l_i	Change in load at location $i \in \mathcal{N}$
u_i	Maximum ride time for customer with pickup at $i \in \mathcal{P}$
C^k	Capacity of vehicle $k \in \mathcal{K}$
Q	Effective battery capacity
B_0^k	Initial battery capacity of vehicle $k \in \mathcal{K}$
r	Final minimum battery level ratio
$\beta_{i,j}$	Battery consumption between nodes $i, j \in \mathcal{V}$
α_s	Recharge rate at charging facility $s \in \mathcal{S}$
T_p	Planning horizon
Decision Variables	
$x_{i,j}^k$	1 if vehicle k sequentially stops at location i and $j \in \mathcal{V}$, 0 otherwise.
T_i^k	Time at which vehicle k starts its service at location $i \in \mathcal{V}$
L_i^k	Load of vehicle k at location $i \in \mathcal{V}$
B_i^k	Battery load of vehicle k at location $i \in \mathcal{V}$
E_s^k	Charging time of vehicle k at charging station $s \in \mathcal{S}$
R_i	Excess ride time of passenger $i \in \mathcal{P}$

A binary decision variable $x_{i,j}^k$ denotes whether vehicle k sequentially visits locations i and $j \in \mathcal{V}$. T_i^k represents the time at which vehicle k begins service at location $i \in \mathcal{V}$, L_i^k represents its load after service, and B_i^k represents its battery state at the beginning of service. In addition, E_s^k denotes the recharge time of vehicle k at station $s \in \mathcal{S}$. Finally, R_i represents the excess time of user $i \in \mathcal{P}$. The objective of the e-ADARP is to find minimum cost routes in order to serve all users while respecting time-window, capacity, and battery constraints. The cost is defined by means of a weighted objective function consisting of the total travel time of all vehicles and excess ride time of all users. A summary of the e-ADARP problem sets, parameters, and decision variables is reported in Table 3.2. Note that the parameters related to battery consumption, recharge rate, and final minimum battery level may be defined as vehicle-specific. Similarly, vehicle-specific charging stations and optional destination depots can be defined. Nevertheless, in the following models we assume that such parameters and sets are homogeneous among all vehicles. This assumption is reasonable in the employment of autonomous electric mini-buses, as nowadays such vehicles mount the same commercial batteries and do not significantly differ in terms of dimensions and weights.

Table 2.2: 3-indexed formulation for the e-ADARP

$$(e-ADARP3) \quad \min w_1 \sum_{k\in\mathcal{K}} \sum_{i,j\in\mathcal{V}} t_{i,j} x^k_{i,j} + w_2 \sum_{i\in\mathcal{P}} R_i \tag{2.1}$$

subject to:

$$\sum_{j\in\mathcal{P}\cup\mathcal{S}\cup\mathcal{F}} x^k_{o^k,j} = 1 \qquad \forall k\in\mathcal{K} \tag{2.2}$$

$$\sum_{j\in\mathcal{F}} \sum_{i\in\mathcal{D}\cup\mathcal{S}\cup\{o^k\}} x^k_{i,j} = 1 \qquad \forall k\in\mathcal{K} \tag{2.3}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{D}\cup\mathcal{S}\cup\{o^k\}} x^k_{i,j} \leq 1 \qquad \forall j\in\mathcal{F}\cup\mathcal{S} \tag{2.4}$$

$$\sum_{\substack{j\in\mathcal{V}\\ j\neq i}} x^k_{i,j} - \sum_{\substack{j\in\mathcal{V}\\ j\neq i}} x^k_{j,i} = 0 \qquad \forall k\in\mathcal{K}, i\in\mathcal{N}\cup\mathcal{S} \tag{2.5}$$

$$\sum_{k\in\mathcal{K}} \sum_{\substack{j\in\mathcal{N}\\ j\neq i}} x^k_{i,j} = 1 \qquad \forall i\in\mathcal{P} \tag{2.6}$$

$$\sum_{\substack{j\in\mathcal{N}\\ j\neq i}} x^k_{i,j} - \sum_{\substack{j\in\mathcal{N}\\ j\neq n+i}} x^k_{j,n+i} = 0 \qquad \forall k\in\mathcal{K}, i\in\mathcal{P} \tag{2.7}$$

$$T^k_i + d_i + t_{i,n+i} \leq T^k_{n+i} \qquad \forall k\in\mathcal{K}, i\in\mathcal{P} \tag{2.8}$$

$$arr_i \leq T^k_i \leq dep_i \qquad \forall k\in\mathcal{K}, i\in\mathcal{V} \tag{2.9}$$

$$T^k_{n+i} - T^k_i - d_i \leq u_i \qquad \forall k\in\mathcal{K}, i\in\mathcal{P} \tag{2.10}$$

$$T^k_i + t_{i,j} + d_i - M_{i,j}(1-x^k_{i,j}) \leq T^k_j \qquad \forall k\in\mathcal{K}, i\in\mathcal{V}, j\in\mathcal{V}, i\neq j | M_{i,j} > 0 \tag{2.11}$$

$$R_i \geq T^k_{n+i} - T^k_i - d_i - t_{i,n+i} \qquad \forall k\in\mathcal{K}, i\in\mathcal{P} \tag{2.12}$$

$$L^k_i + l_j - G^k_{i,j}(1-x^k_{i,j}) \leq L^k_j \qquad \forall k\in\mathcal{K}, i\in\mathcal{V}, j\in\mathcal{V}, i\neq j \tag{2.13}$$

$$L^k_i + l_j + G^k_{i,j}(1-x^k_{i,j}) \geq L^k_j \qquad \forall k\in\mathcal{K}, i\in\mathcal{V}, j\in\mathcal{V}, i\neq j \tag{2.14}$$

$$L^k_i \geq \max(0, l_i) \qquad \forall k\in\mathcal{K}, \forall i\in\mathcal{N} \tag{2.15}$$

$$L^k_i \leq \min(C^k, C^k + l_i) \qquad \forall k\in\mathcal{K}, \forall i\in\mathcal{N} \tag{2.16}$$

$$L^k_i = 0 \qquad \forall k\in\mathcal{K}, i\in o^k\cup\mathcal{F}\cup\mathcal{S} \tag{2.17}$$

$$B^k_i = B^k_0 \qquad \forall k\in\mathcal{K}, i\in o^k \tag{2.18}$$

$$B^k_j \leq B^k_i - \beta_{i,j} + Q(1-x^k_{i,j}) \qquad \forall k\in\mathcal{K}, i\in\mathcal{V}\setminus\mathcal{S}, j\in\mathcal{V}\setminus\{o^k\}, i\neq j \tag{2.19}$$

$$B^k_j \geq B^k_i - \beta_{i,j} - Q(1-x^k_{i,j}) \qquad \forall k\in\mathcal{K}, i\in\mathcal{V}\setminus\mathcal{S}, j\in\mathcal{V}\setminus\{o^k\}, i\neq j \tag{2.20}$$

$$B^k_j \leq B^k_s + \alpha_s E^k_s - \beta_{s,j} + Q(1-x^k_{s,j}) \qquad \forall k\in\mathcal{K}, s\in\mathcal{S}, j\in\mathcal{P}\cup\mathcal{F}\cup\mathcal{S}, s\neq j \tag{2.21}$$

$$B^k_j \geq B^k_s + \alpha_s E^k_s - \beta_{s,j} - Q(1-x^k_{s,j}) \qquad \forall k\in\mathcal{K}, s\in\mathcal{S}, j\in\mathcal{P}\cup\mathcal{F}\cup\mathcal{S}, s\neq j \tag{2.22}$$

$$Q \geq B^k_s + \alpha_s E^k_s \qquad \forall k\in\mathcal{K}, s\in\mathcal{S} \tag{2.23}$$

$$B^k_i \geq rQ \qquad \forall k\in\mathcal{K}, i\in\mathcal{F} \tag{2.24}$$

$$E^k_s \leq T^k_s - t_{i,s} - T^k_i + M^k_{i,s}(1-x^k_{i,s}) \qquad \forall k\in\mathcal{K}, \forall s\in\mathcal{S}, i\in\mathcal{D}\cup\mathcal{S}\cup\{o^k\}, i\neq s \tag{2.25}$$

$$E^k_s \geq T^k_s - t_{i,s} - T^k_i - M^k_{i,s}(1-x^k_{i,s}) \qquad \forall k\in\mathcal{K}, \forall s\in\mathcal{S}, i\in\mathcal{D}\cup\mathcal{S}\cup\{o^k\}, i\neq s \tag{2.26}$$

$$x^k_{i,j} \in \{0,1\} \qquad \forall k\in\mathcal{K}, i\in\mathcal{V}, j\in\mathcal{V} \tag{2.27}$$

$$B^k_i \geq 0 \qquad \forall k\in\mathcal{K}, i\in\mathcal{V} \tag{2.28}$$

$$E^k_s \geq 0 \qquad \forall k\in\mathcal{K}, s\in\mathcal{S} \tag{2.29}$$

A 3-index formulation for the e-ADARP, building on the DARP formulation provided in Cordeau (2006), is defined by the MILP presented in Table 2.2. The objective function (2.1) minimized a weighted-sum composed of the total vehicle travel time and user excess ride time. Constraints (2.2) ensure that all vehicles exit their origin depots and visit a pickup location $\mathcal{P}$, relocate to a charging station $\mathcal{S}$, or proceed to a destination depot $\mathcal{F}$. Constraints (2.3) guarantee that all vehicles return to a selected destination depot. Note that, since $\mathcal{F}$ also includes the origin depots $\mathcal{O}$, a non-utilized vehicle travels between its origin depots $o^k \in \mathcal{O}$ and its geographically coincident destination depot $\bar{o}^k \in \mathcal{F}$. Therefore, cases where $x^k_{o^k,\bar{o}^k} = 1$ imply that vehicle k is not being used. Constraints (2.4) allow each optional destination depot and charging station to be visited at most once. Such locations can be replicated in order to allow multiple visits to the nodes in $\mathcal{F}$ and $\mathcal{S}$. Flow conservation is ensured by constraints (2.5). Constraints (2.6)-(2.7) ensure that every pickup location is visited exactly once and that each pickup and dropoff pair is served by the same vehicle.

Timing constraints are added to define service start times and excess ride times. Though not explicitly denoted by decision variables, waiting can occur at any location and can be retrieved through post-processing. Constraints (2.8) guarantee that each pickup location i is visited before its dropoff location $n+i$, by means of the direct travel time between the two locations and service time at location i. Constraints (2.9) set time windows around the beginning of service at each location and constraints (2.10) impose maximum ride times for the users. Constraints (2.11) set a lower bound on the service start time at location $j \in \mathcal{V}$, which is visited right after location $i \in \mathcal{V}$, where $M_{i,j} = \max\{0, dep_i + d_i + t_{i,j} - arr_j\}$. Constraints (2.12) calculate the excess ride time for user i, where the ideal travel time $t_{i,n+i} + d_i$ of user $i \in \mathcal{P}$ is subtracted from the actual travel time $T^k_{n+i} - T^k_i$.

Load conservation and capacity constraints are added to track loads and ensure capacities of all vehicles are not exceeded. We assume that charging stations can only be visited when vehicles are empty. In constraints (2.13)-(2.14), the load at location $j \in \mathcal{V}$ is computed from the load at the preceding location $i \in \mathcal{V}$ and the change in load at location j, where $G^k_{i,j} = \min\{C^k, C^k + l_i\}$. Constraints (2.14) are redundant with respect to the model formulation. However, they help strengthening the LP relaxations. Constraints (2.15)-(2.16) set lower and upper bounds on the occupancy of all vehicles and constraints (2.17) ensure that vehicles are empty at depots and charging stations.

Battery-management constraints are introduced to track battery levels when traveling between locations and during recharge phases. Furthermore, vehicles have initial battery levels and need to have minimum battery levels at the end of the planning horizon. Constraints (2.18) set initial battery levels for vehicles at origin depots o^k. Constraints (2.19)-(2.20) set the battery level state from any location $i \in V \setminus \mathcal{S}$ to any location $j \in V \setminus o(k)$, while constraints (2.21)-(2.22) set the battery level state after a visit to a charging facility $s \in \mathcal{S}$ to any location $j \in \mathcal{P} \cup \mathcal{F} \cup \mathcal{S}$. Constraints (2.23) set upper bounds on the battery level states at charging stations while constraints (2.24) impose minimum battery levels for all vehicles returning to the depots. Constraints (2.25)-(2.26) define upper and lower bounds on the recharge time at charging station $s \in S$. Integrality and non-negativity constraints are set in (2.27)-(2.29).

2.2.2 Two-Index Formulation

In this section we present a 2-index mathematical formulation for the e-ADARP. Note that, although 2-index-based branch-and-cut algorithms typically outperform 3-indexed frameworks (Ropke, Cordeau, and Laporte, 2007; Parragh, 2011; Braekers, Caris, and Janssens, 2014), a 3-index formulation for the e-ADARP becomes necessary when considering vehicle-dependent battery consumptions and recharge rates.

The main limitation when transforming a 3-index formulation into a 2-index formulation is that the decision variables do not feature an index k representing the vehicles and, as a result, vehicle-specific constraints can no longer be directly modeled. Nevertheless, violations of pairing, precedence, and capacity considerations can be prevented by introducing a set of exponential constraints which are then separated during the search, as first presented by Ruland and Rodin (1997) and later applied by Ropke, Cordeau, and Laporte (2007), Parragh (2011), and Braekers, Caris, and Janssens (2014). Both multiple depots and heterogeneity in terms of capacity and initial battery inventories can be modeled by considering a singular common artificial origin and destination depot $\{0, 2n+1\}$ and by representing $\mathcal{O}$ and $\mathcal{F}$ as dummy origin and destination depots respectively (Baldacci, Battarra, and Vigo, 2009; Parragh, 2011; Braekers, Caris, and Janssens, 2014). As a result, in the 2-index formulation, the vertex set $\mathcal{V}$ consists of sets $\{0, 2n+1\}, \mathcal{P}, \mathcal{D}, \mathcal{O}, \mathcal{F}, \mathcal{S}$. As in Braekers, Caris, and Janssens (2014) several arcs (i, j) are eliminated from the arc set $\mathcal{V}$ to ensure that each route starts with a dummy origin depot and ends at a dummy destination depot. Therefore, the following arcs (i, j) are eliminated from the complete graph $\mathbb{G}$:
$(i, 0), (2n+1, i) \quad \forall i \in \mathcal{V}$
$(0, j) \quad \forall j \in \mathcal{V} \setminus \mathcal{O}$
$(i, 2n+1) \quad \forall i \in \mathcal{V} \setminus \mathcal{F}$

To impose precedence and pairing constraints, $\mathbb{I}$ denotes the set of all vertex subsets $\mathcal{I} \subseteq \mathcal{V} \setminus \mathcal{S}$ such that $0 \in \mathcal{I}, 2n+1 \notin \mathcal{I}$ and there is at least one vertex i for which $i \notin \mathcal{I}$ and $n+i \in \mathcal{I}$. $\bar{\mathbb{I}}$ denotes the complementary set of $\mathbb{I}$ with $\mathcal{S} \not\subset \bar{\mathcal{I}}$, $\forall \bar{\mathcal{I}} \in \mathbb{I}$. Then, given precedence and paring considerations between i and $n+i$, sets $\mathcal{I} \in \mathbb{I}$ need to be exited at least once. Given that charging stations are not included in $\mathbb{I}$ or $\bar{\mathbb{I}}$, and that at least one arc needs to directly exit $\mathcal{I} \in \mathbb{I}$ towards $\bar{\mathcal{I}} \in \bar{\mathbb{I}}$, precedence and pairing constraints also prevent non-empty vehicles from entering charging stations $\mathcal{S}$. Capacity constraints can be enforced by setting a lower bound on the number of times vehicles must enter and exit $\mathcal{I}$ by computing $\max\{1, \lceil \frac{|\sum_{i \in \mathcal{I}} l_i|}{C} \rceil\}$ (Ropke, Cordeau, and Laporte, 2007).

After visiting a dummy depot, capacities are set such that they correspond to their actual levels, as presented in Braekers, Caris, and Janssens (2014). Therefore $l_i = C - C^k \ \forall i \in \mathcal{O}$ and $l_i = C^k - C \ \forall i \in \mathcal{F}$. The vehicle battery levels are instead set to the initial battery inventories B_{o_k} at every origin depot $o_k \in \mathcal{O}$.

Table 2.3: 2-indexed formulation for the e-ADARP

$$(e-ADARP2) \quad \min w_1 \sum_{i,j \in \mathcal{V}} t_{i,j} x_{i,j} + w_2 \sum_{i \in \mathcal{P}} R_i \tag{2.30}$$

subject to:

$$\sum_{j \in \mathcal{O}} x_{0,j} = |\mathcal{K}| \tag{2.31}$$

$$\sum_{i \in \mathcal{F}} x_{i,2n+1} = |\mathcal{K}| \tag{2.32}$$

$$\sum_{i \in \mathcal{D} \cup \mathcal{S} \cup \mathcal{O}} x_{i,j} \leq 1 \qquad \forall j \in \mathcal{F} \cup \mathcal{S} \tag{2.33}$$

$$\sum_{j \in \mathcal{N}} x_{i,j} = 1 \qquad \forall i \in \mathcal{P} \tag{2.34}$$

$$\sum_{i \in \mathcal{N}} x_{i,j} = 1 \qquad \forall j \in \mathcal{D} \tag{2.35}$$

$$\sum_{j \in \mathcal{V}} x_{i,j} - \sum_{j \in \mathcal{V}} x_{j,i} = 0 \qquad \forall i \in \mathcal{N} \cup \mathcal{S} \cup \mathcal{O} \tag{2.36}$$

$$\sum_{i,j \in \mathcal{I}} x_{i,j} \leq |\mathcal{I}| - 2 \qquad \forall \mathcal{I} \in \mathbb{I} \tag{2.37}$$

$$\sum_{i,j \in \mathcal{I}} x_{i,j} \leq |\mathcal{I}| - \max\{1, \lceil \frac{|\sum_{i \in \mathcal{I}} l_i|}{C} \rceil\} \qquad \forall \mathcal{I} \in \mathbb{I} \tag{2.38}$$

$$T_i + d_i + t_{i,n+i} \leq T_{n+i} \qquad \forall i \in \mathcal{P} \tag{2.39}$$

$$arr_i \leq T_i \leq dep_i \qquad \forall i \in \mathcal{V} \tag{2.40}$$

$$T_{n+i} - T_i - d_i \leq u_i \qquad \forall i \in \mathcal{P} \tag{2.41}$$

$$T_i + t_{i,j} + d_i - M_{i,j}(1 - x_{i,j}) \leq T_j \qquad \forall i \in \mathcal{V}, j \in \mathcal{V}, i \neq j | M_{i,j} > 0 \tag{2.42}$$

$$R_i \geq T_{n+i} - T_i - d_i - t_{i,n+i} \qquad \forall i \in \mathcal{P} \tag{2.43}$$

$$B_j \leq B_i - \beta_{i,j} + Q(1 - x_{i,j}) \qquad \forall i \in \mathcal{V} \setminus \mathcal{S} \cup \{0\}, j \in \mathcal{V} \setminus \mathcal{O} \cup \{0\}, i \neq j \tag{2.44}$$

$$B_j \geq B_i - \beta_{i,j} - Q(1 - x_{i,j}) \qquad \forall i \in \mathcal{V} \setminus \mathcal{S} \cup \{0\}, j \in \mathcal{V} \setminus \mathcal{O} \cup \{0\}, i \neq j \tag{2.45}$$

$$B_j \leq B_s + \alpha_s E_s - \beta_{s,j} + Q(1 - x_{s,j}) \qquad \forall s \in \mathcal{S}, j \in \mathcal{P} \cup \mathcal{F} \cup \mathcal{S}, s \neq j \tag{2.46}$$

$$B_j \geq B_s + \alpha_s E_s - \beta_{s,j} - Q(1 - x_{s,j}) \qquad \forall s \in \mathcal{S}, j \in \mathcal{P} \cup \mathcal{F} \cup \mathcal{S}, s \neq j \tag{2.47}$$

$$E_s \leq T_s - t_{i,s} - T_i + M_{i,s}(1 - x_{i,s}) \qquad \forall s \in \mathcal{S}, i \in \mathcal{D} \cup \mathcal{S} \cup \mathcal{O}, i \neq s \tag{2.48}$$

$$E_s \geq T_s - t_{i,s} - T_i - M_{i,s}(1 - x_{i,s}) \qquad \forall s \in \mathcal{S}, i \in \mathcal{D} \cup \mathcal{S} \cup \mathcal{O}, i \neq s \tag{2.49}$$

$$Q \geq B_s + \alpha_s E_s \qquad \forall s \in \mathcal{S} \tag{2.50}$$

$$B_i \geq rQ \qquad \forall i \in \mathcal{F} \tag{2.51}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall i \in \mathcal{V}, j \in \mathcal{V} \tag{2.52}$$

$$B_i \geq 0 \qquad \forall i \in \mathcal{V} \tag{2.53}$$

$$E_s \geq 0 \qquad \forall s \in \mathcal{S} \tag{2.54}$$

The 2-indexed e-ADARP can be formulated as the MILP presented in Table 2.3. Note that the second criterion in the objective function (2.30) depends on scheduling decisions. Therefore, timing decision variables and constraints cannot be omitted from the 2-index model for the e-ADARP.

Differently from the 3-index model presented in the previous section, the artificial origin and destination depots $\{0, 2n+1\}$ are exited/entered $|\mathcal{K}|$ times. As with the charging stations $\mathcal{S}$, each node in $\mathcal{F}$ may be visited at most once, as shown in constraints (2.33). Nevertheless, the interaction with constraints (2.32) and constraints (2.36) impose that at least $|\mathcal{K}|$ arcs enter and exit $\mathcal{F}$. Precedence and pairing constraints are imposed by inequalities (2.37), whereas rounded capacity constraints are given in (2.38). As noted in Ropke, Cordeau, and Laporte (2007), since $\sum\limits_{i,j \in \mathcal{I}} x_{i,j} = |\mathcal{I}| - 1 - \sum\limits_{i \in \mathcal{I}} \sum\limits_{i \in \bar{\mathcal{I}}} x_{i,j}$, precedence constraints (2.37) can be equivalently written as $\sum\limits_{i \in \mathcal{I}} \sum\limits_{i \in \bar{\mathcal{I}}} x_{i,j} \geq 1 \ \forall \mathcal{I} \in \mathbb{I}$. Finally, given that each node can be visited by one vehicle only, the remaining constraints (2.39)-(2.54) are equivalent to constraints (2.8)-(2.29).

2.3 Valid Inequalities

In this section, we introduce several families of valid inequalities used in order to strengthen the e-ADARP formulations. In Section 2.3.1, we present sets of valid inequalities that are adopted from the vehicle routing literature. Then, in Section 2.4, we discuss several sets of valid inequalities that are specifically designed by taking into account properties of the e-ADARP.

2.3.1 Valid Inequalities from the Vehicle Routing Literature

Several families of valid inequalities from the vehicle routing literature are also valid for the e-ADARP and can be lifted to account for the charging stations. In particular, the following inequalities can be directly applied to the e-ADARP: (1) time-window strengthening (Cordeau, 2006), (2) incompatible users constraints (Cordeau, 2006), (3) generalized order constraints (Ruland and Rodin, 1997;Cordeau, 2006), (4) subtour elimination constraints (Cordeau, 2006) (Note that subtours can also include the charging stations), (5) strengthened capacity constraints (Ropke, Cordeau, and Laporte, 2007), (6) infeasible path constraints based on maximum ride time considerations (Cordeau, 2006), (7) tournament constraints (Ascheuer, Fischetti, and Grötschel, 2000, Ropke, Cordeau, and Laporte, 2007), (8) fork constraints (Ropke, Cordeau, and Laporte, 2007), and (9) reachability constraints (Lysgaard, 2006; Ropke, Cordeau, and Laporte, 2007).

In the case of the e-ADARP, capacity constraints can be lifted considering the recharge stations. This set of constraints is added to prevent situations in which the number of vehicles visiting set $\mathbb{S} \subseteq \mathcal{N}$ is not sufficient to accommodate its demand. Considering $C = \max\limits_{k \in \mathcal{K}} C$, and letting $\mathbb{S}, \mathbb{T} \subseteq \mathcal{N}$ be two disjoint sets such that $\sum_{i \in \mathbb{S}} l_i > 0$, and $\mathbb{U} = \pi(\mathbb{T}) \setminus (\mathbb{S} \cup \mathbb{T})$, where $\pi(\mathbb{T}) = \{i \in \mathcal{P} | n+i \in \mathbb{T}\}$, capacity constraints can be written as follows (Ropke, Cordeau, and Laporte, 2007):

$$x(\mathbb{S}) + x(\mathbb{T}) + x(\mathbb{S} : \mathbb{T}) \leq |\mathbb{S}| + |\mathbb{T}| - \left\lceil \frac{l(\mathbb{S}) + l(\mathbb{U})}{C} \right\rceil \tag{2.55}$$

In the case of the e-ADARP, charging stations can only be accessed when

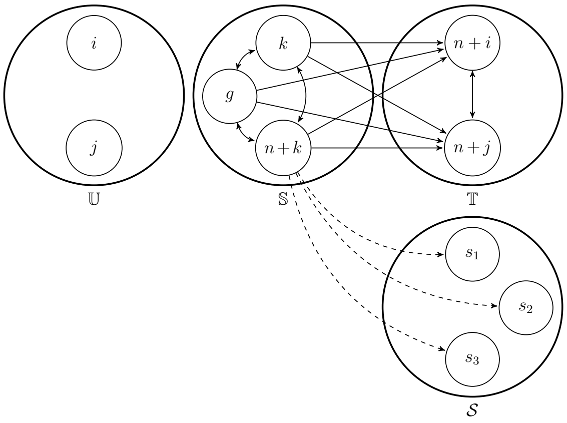


Figure 2.1: Inequalities (2.56) for the case in which $\mathbb{U} = \{i, j\}$, $\mathbb{S} = \{k, g, n + k\}$, $\mathbb{T} = \{n + i, n + j\}$, and $\mathcal{S} = \{s_1, s_2, s_3\}$

vehicles are unloaded (2.17) and each node can be exited at most once. Therefore, capacity constraints can be lifted considering arcs from dropoff locations in $\mathbb{S}$ to the set of stations $\mathcal{S}$, as follows:

$$x(\mathbb{S}) + x(\mathbb{T}) + x(\mathbb{S} : \mathbb{T}) + x((\mathbb{S} \cap \mathcal{D}) : \mathcal{S}) \leq |\mathbb{S}| + |\mathbb{T}| - \left\lceil \frac{l(\mathbb{S}) + l(\mathbb{U})}{C} \right\rceil \tag{2.56}$$

Figure 2.1 shows an illustrative example of inequalities (2.56) for the case in which $\mathbb{U} = \{i, j\}$, $\mathbb{S} = \{k, g, n + k\}$, $\mathbb{T} = \{n + i, n + j\}$, and $\mathcal{S} = \{s_1, s_2, s_3\}$.

2.4 Valid Inequalities for the e-ADARP

The following valid inequalities are based on battery-management aspects introduced in the e-ADARP. In Section 2.4.1 and Section 2.4.2 we present infeasible path constraints that consider battery capacity and load restrictions at charging stations respectively. We note that in both cases, the resulting constraints can be presented in the structure of tournament constraints and used in the generation of fork constraints (Ropke, Cordeau, and Laporte, 2007).

2.4.1 Infeasible Path Inequalities – Battery Capacity

Similar to the maximum ride time constraints introduced in Cordeau (2006), it is possible to introduce a set of infeasible path inequalities related to battery-management. The vehicles considered in this study are electric. As such, they need to have enough battery to perform partial paths without recharge. Consider a path $\mathbb{P} = \{s', i, m_1, m_2, \ldots, m_q, n + i, s''\}$, such that $\mathcal{H} = \{m_1, m_2, \ldots, m_q\} \subseteq \mathcal{N}$, $i \in \mathcal{P}$, and $\{s', s''\} \in \mathcal{S}$. In terms of battery consumption, s' is the closest charging station to location i and s'' is the closest station to $n + i$, i.e. $s' = \arg\min_{s \in \mathcal{S}} \beta_{s,i}$ and $s'' = \arg\min_{s \in \mathcal{S}} \beta_{n+i,s}$. Assuming that the triangular inequality for battery consumption holds, and considering the effective battery capacity Q, we deem path $\mathbb{P}$ to be infeasible if the battery consumption required to traverse it is greater than Q. That is, if $\beta_{s',i} + \beta_{i,m_1} + \sum_{h=1}^{q-1} \beta_{m_h,m_{h+1}} + \beta_{m_p,n+i} + \beta_{n+i,s''} > Q$.

Next, vehicles need to return to destination depots having some minimum battery levels, which depend on the minimal battery level ratio r. Then, assuming that the destination depot $f \in \mathcal{F}$ is chosen such that the battery consumption from $n+i$, i.e. $f = \arg\min_{f \in \mathcal{F}} \beta_{n+i,f}$, is minimized if $\beta_{s',i}+\beta_{i,m_1}+\sum_{h=1}^{q-1} \beta_{m_h,m_{h+1}}+\beta_{m_p,n+i}+\beta_{n+i,f} > (1-r)Q$, path $\mathbb{P}' = \{s', i, m_1, m_2, \dots, m_q, n+i, f\}$ is infeasible.

Furthermore, vehicles start their service with some initial battery level B_{o_k}. Assuming the maximum initial SOC is set to $B_{o_{max}} = \max_{k \in \mathcal{K}} B_{o_k}$, and the origin depot $o_k \in \mathcal{O}$ is chosen such that it minimizes battery consumption to i, i.e. $o = \arg\min_{o_k \in \mathcal{O}} \beta_{o_k,i}$. Then, if $\beta_{o,i} + \beta_{i,m_1} + \sum_{h=1}^{q-1} \beta_{m_h,m_{h+1}} + \beta_{m_q,n+i} + \beta_{n+i,s''} > B_{o_{max}}$, path $\mathbb{P}'' = \{o, i, m_1, m_2, \dots, m_q, n+i, s''\}$ is infeasible.

Finally, let $\mathcal{A}(\mathcal{H})$ be the edges of $\mathcal{H}$. Then, if all paths $\mathbb{P}$, $\mathbb{P}'$, and $\mathbb{P}''$ are infeasible, path $(i, \mathcal{H}, n+i)$ can be eliminated by the following inequalities (Cordeau, 2006):

$$x_{i,m_1} + \sum_{h=1}^{q-1} x_{m_h,m_{h+1}} + x_{m_q,n+i} \leq q-1 \tag{2.57}$$

2.4.2 Infeasible Path Inequalities – Charging Stations “Walls"

Recall that, in the e-ADARP, vehicles are not allowed to visit charging stations with passengers on board (constraints 2.17). Therefore, any visit to a charging station between the pickup of a customer and its dropoff is infeasible. Consider a path $\mathbb{P}'$ connecting the pickup location of user i and a charging station s by a sequence of q intermediate locations $\{m_1, m_2, \dots, m_q\}$ with $m_{1,\dots,q} \in \mathcal{P} \cup \mathcal{D} \setminus \{n+i\}$. Note that, in any feasible integer solution, visits to charging stations are only possible when vehicles are empty and, as such, at most q arcs can be selected from path $\mathbb{P}'$. These inequalities can be further lifted by including arcs connecting the charging stations $s \in \mathcal{S}$ to the last intermediate location m_q, and from m_q to s, as follows:

$$x_{i,m_1} + \sum_{h=1}^{q-1} x_{m_h,m_{h+1}} + \sum_{s \in \mathcal{S}} (x_{m_q,s} + x_{s,m_q}) \leq q \tag{2.58}$$

An illustrative example for inequalities (2.58) for the case in which $\mathbb{P} = \{i, m_1, m_2, \dots, m_q, s_2\}$ and $|\mathcal{S}| = 3$ is given in Figure 2.2.

Similarly, infeasible path inequalities (2.59) are derived by considering a path $\mathbb{P}''$ connecting a charging station s to a dropoff location $n+i$ by a sequence of q intermediate locations $\{m_1, m_2, \dots, m_q\}$ with $m_{1,\dots,q} \in \mathcal{D} \cup \mathcal{P} \setminus \{i\}$.

$$\sum_{s \in \mathcal{S}} (x_{s,m_1} + x_{m_1,s}) + \sum_{h=1}^{q-1} x_{m_h,m_{h+1}} + \sum_{s \in \mathcal{S}} x_{m_q,n+i} \leq q \tag{2.59}$$

An illustrative example for inequalities (2.59) for the case in which $\mathbb{P}'' = \{s_2, m_1, m_2, \dots, m_q, n+i\}$ and $|\mathcal{S}| = 3$ is given in Figure 2.3.

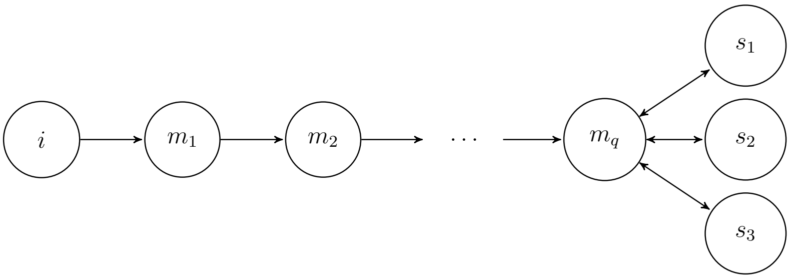


Figure 2.2: Inequalities (2.58) for the case in which $\mathbb{P} = \{i, m_1, m_2, \ldots, m_q, s_2\}$ and $|\mathcal{S}| = 3$

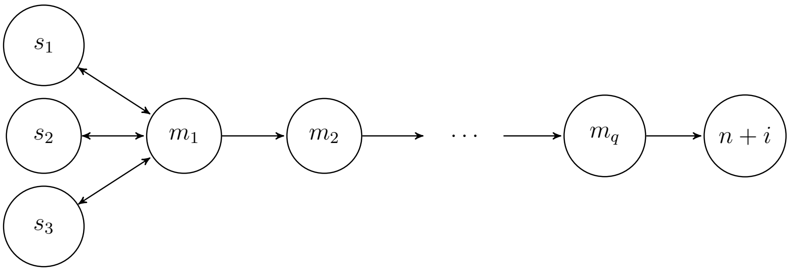


Figure 2.3: Inequalities (2.59) for the case in which $\mathbb{P}'' = \{s_2, m_1, m_2, \ldots, m_1, n+i\}$ and $|\mathcal{S}| = 3$

2.4.3 Charging Stations and Time Windows Constraints

We derive a set of constraints that reflect the interaction between charging stations and time windows. Suppose a vehicle leaves by the earliest time from $n+i \in \mathcal{D}$, recharges at $s \in \mathcal{S}$, and travels to $j \in \mathcal{P}$. Then, in order to justify the deviation from $n+i$ to s before reaching j, the vehicle needs to recharge for a minimum time. This time can be inferred from the battery consumptions between $n+i$, s, j and the recharge rate α_s, as $\frac{\beta_{n+i,s}+\beta_{s,j}-\beta_{n+i,j}}{\alpha_s}$. Then, if path $\{n+i, s, j\}$ is infeasible with respect to its start and end time windows, i.e. $t_{n+i,s}+d_{n+i}+t_{s,j}+\frac{\beta_{n+i,s}+\beta_{s,j}-\beta_{n+i,j}}{\alpha_s} > dep_j - arr_{n+i}$, the following infeasible path inequalities are identified:

$$x_{n+i,s} + x_{s,j} + x_{n+j,s} \leq 1 \quad \forall i \in \mathcal{P}, j \in \mathcal{N}$$

Next, suppose the same vehicle travels from $n+j$ to station s and then reaches i. In this case, compute a minimum recharge time $\frac{\beta_{n+j,s}+\beta_{s,i}-\beta_{n+j,i}}{\alpha_s}$ at station s. Then, if both paths $\{n+i, s, j\}$ and $\{n+j, s, i\}$ are violated, i.e. $t_{n+i,s} + d_{n+i} + t_{s,j} + \frac{\beta_{n+i,s}+\beta_{s,j}-\beta_{n+i,j}}{\alpha_s} > dep_j - arr_{n+i}$ and $t_{n+j,s} + d_{n+j} + t_{s,i} + \frac{\beta_{n+j,s}+\beta_{s,i}-\beta_{n+j,i}}{\alpha_s} > dep_i - arr_{n+j}$, stronger inequalities can be added as follows:

$$x_{n+i,s} + x_{s,j} + x_{n+j,s} + x_{s,i} \leq 1 \quad \forall i \in \mathcal{P}, j \in \mathcal{N} \tag{2.60}$$

An illustrative example for inequalities (2.60) is given in Figure 2.4.

2.4.4 Charging Visits Constraint

In the e-ADARP, one can set a lower bound on the minimal number of visits to charging stations by considering the total amount of battery consumed by all

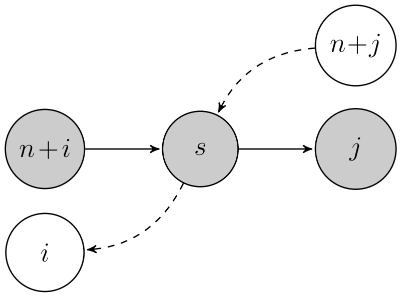


Figure 2.4: Inequalities (2.60): Charging stations and time windows constraints

vehicles during service. On the left hand-side of inequality (2.61), the number of times the vehicles fully discharge with respect to the effective battery capacity Q is computed. Note each vehicle starts and ends service with positive battery levels. For this reason, the total maximal initial SOC $B_{o_{max}}$ for all utilized vehicles (i.e. all vehicles k with $x_{o^k,\bar{o}^k} = 0$) is deducted from the left-hand side, while the total final SOC for all utilized vehicles is added back to the left-hand-side of inequality (2.61). On the right-hand side of inequalities (2.61), the sum represents the minimum number of times all vehicles exit charging stations, i.e. the minimum number of visits to charging stations.

$$\sum_{i,j\in\mathcal{V},i\neq j}\frac{\beta_{i,j}x_{i,j}}{Q} - (|\mathcal{K}| - \sum_{i\in\mathcal{O}}\sum_{j\in\bigcup\limits_{k\in\mathcal{K}}\{\bar{o}^k\}} x_{i,j})(\frac{B_{o_{max}}}{Q} - r) \leq \sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{P}\cup\mathcal{S},i\neq s} x_{s,i} \tag{2.61}$$

Note that inequality (2.61) is inspired by a similar concept in the context of the locomotive refuelling problem (Raviv and Kaspi, 2012). The constraint is also similar to the rounded capacity constraints introduced in Cordeau, 2006 where, in place of passenger capacity and loads, total battery consumption and visits to stations are considered. Finally, note that for the 3-index model, inequality (2.61) can be vehicle-specific.

2.5 Branch-and-Cut Algorithm

The following sections present the branch-and-cut framework developed for the e-ADARP. In Section 2.5.1, we discuss the steps taken at the initialization of the algorithm. These include pre-processing procedures and the enumeration of several sets of inequalities. In Section 2.5.2, we describe the separation heuristics designed to identify violated valid inequalities as well as capacity, precedence, and pairing constraints (2.37)-(2.38).

2.5.1 Initialization

In order to reduce the size of the problem and strengthen the LP relaxation, several steps are taken before solving the root node. These steps include time-window tightening, arc elimination, variable fixing, symmetry breaking, and the enumeration of several sets of valid inequalities. Most of these steps are based on considerations arising from standard DARP properties. For brevity, we omit the description of these initialization steps and refer the reader to Sections 5.1 and 5.2 in Cordeau (2006). Note that precedence, paring, and capacity constraints (2.37)-(2.38) may be also included at the root node by means of the separation heuristics described in Section 2.5.2. In the following paragraphs, we focus our discussion on considerations related to charging stations.

By definition, vehicles are required to enter charging stations with no users on board. Consequently, a visit to a charging station cannot be preceded by a pickup node and cannot be followed by a dropoff node. Specifically, arcs (i, s) with $i \in \mathcal{P}$ and $s \in \mathcal{S}$ and arcs (s, j) with $j \in \mathcal{D}$ are eliminated, as follows:

$$(i, j) \quad \forall i \in \mathcal{O} \cup \mathcal{S}, j \in \mathcal{D}$$

$$(i, j) \quad \forall i \in \mathcal{P}, j \in \mathcal{F} \cup \mathcal{S}$$

Next, charging stations may be visited at any time. That is, the time windows imposed at charging stations include the entire planning horizon. Some minor tightening of these time windows may be achieved by taking into account traveling times from the origin and to the destination depots. Namely, the earliest time to start service at charging station s can be set to $arr_s = \min\limits_{o_k \in \mathcal{O}}(arr_{o_k} + t_{o_k,s})$, and similarly, the latest time to start service at charging station s can be set to $\max\limits_{f \in \mathcal{F}}(T_p - t_{s,f})$. Finally, symmetry breaking constraints can be introduced when stations are replicated to allow for more than a single charging visit. In particular, a visit is allowed only if all preceding visits have been made. Namely, denoting by $\bar{\mathcal{S}}$ the original set of stations nodes and by m the number of station visit replications, the following symmetry breaking constraints are introduced:

$$\sum_{i \in \mathcal{O} \cup \mathcal{S} \cup \mathcal{D}} x_{i,s+j|\bar{\mathcal{S}}|} \geq \sum_{i \in \mathcal{O} \cup \mathcal{S} \cup \mathcal{D}} x_{i,s+(j+1)|\bar{\mathcal{S}}|} \quad \forall s \in \bar{\mathcal{S}}, j = \{0, \ldots, (m-1)\} \tag{2.62}$$

Next, we describe some of the valid inequalities proposed in Section 2.3 for the e-ADARP, which are enumerated during the pre-processing step. The infeasible path constraints presented in Section 2.4.3 are of polynomial size and are fully enumerated at the root node. The charging visits constraint presented in Section 2.4.4 is also inserted at the root node. In addition, we introduce special cases of inequalities (2.58) and (2.59). Specifically, we enumerate the following sets, considering paths $\{i, j, s\}$, where $i \in \mathcal{P}$, $j \in \mathcal{D} \setminus \{n+i\}$, $s \in \mathcal{S}$:

$$x_{i,j} + \sum_{s = \mathcal{S}} x_{j,s} \leq 1 \quad \forall i \in \mathcal{P}, j \in \mathcal{D}, j \neq n+i$$

And, considering paths $\{s, j, n+i\}$, where $i \in \mathcal{P}$, $j \in \mathcal{P} \setminus \{i\}$, $s \in \mathcal{S}$:

$$\sum_{s \in \mathcal{S}} x_{s,j} + x_{j,n+i} \leq 1 \quad \forall i \in \mathcal{P}, j \in \mathcal{P}, j \neq i$$

Finally, for every user $i \in \mathcal{P}$ and charging station $s \in \mathcal{S}$, we generate infeasible path constraints, as follows:

$$x_{n+i,s} + x_{s,i} \leq 1 \quad \forall i \in \mathcal{P}, s \in \mathcal{S}$$

2.5.2 Separation Heuristics

In this Section we present the separation heuristics used in the branch-and-cut algorithm. Precedence and pairing constraints (2.37) are separated using similar separation heuristics as those presented in Parragh (2011), where the Ford-Fulkerson algorithm is implemented to solve the max-flow problems. Rounded capacity constraints (2.38) are detected through an enumerative procedure: from every origin depot in $\mathcal{O}$ we extend paths along the arcs with the maximum flow $x_{i,j}$ until capacity is violated or one destination depot has been reached.

To search for violations of generalized order constraints and strengthened capacity constraints (2.55)-(2.56), we adopt the separation heuristics described in Section 4 in Ropke, Cordeau, and Laporte (2007). Reachability constraints are also separated as per Ropke, Cordeau, and Laporte (2007). Note that, in the e-ADARP sets A_i^+ and A_i^- also need to include arcs to/from charging stations. Moreover, since nodes $s \in \mathcal{S}$ can only be accessed by empty vehicles, sets A_i^+ and A_i^- do not contain arcs from pickups to charging stations or from stations to dropoff locations.

Subtour elimination constraints, tournament constraints with respect to time windows, maximum ride time, and battery-management constraints are separated by the use of a greedy construction heuristic. For each pickup node $i \in \mathcal{P}$, we initialize a path with node i and iteratively append nodes to the path, such that the value of the added arc is maximized. This iterative procedure is terminated when one of the following conditions are met: (1) the next node to be appended is a destination depot $f \in \mathcal{F}$, (2) the next node to be appended already exists in the path, or (3) the path length exceeds a pre-defined value. If the construction of a path is terminated due to a repetition of a node, a subtour may have been identified. As such, in this instance, we check for violations of the subtour elimination constraints introduced in Cordeau (2006). During the construction of the path, we check both whether time-window constraints are violated for the last node appended to the path and whether the value of the relaxed solution violates tournament constraints (Ropke, Cordeau, and Laporte, 2007). If a tournament constraint is not identified (i.e. the backbone path is feasible), we search for violations of fork constraints using the search heuristics presented in Ropke, Cordeau, and Laporte (2007). Infeasible path inequalities with respect to maximum ride time and battery-management constraints are examined only when the constructed path includes the dropoff node $n + i$ of seed node i and does not include any charging station. For such paths, if the travel time exceeds the maximum ride time of user i, or the three battery consumption conditions presented in Section 2.4.1 are met, we check for violations of inequalities (2.57).

Infeasible paths due to load considerations at the charging stations (2.58)-(2.59) are identified by using a heuristic which iteratively considers a node $s \in \mathcal{S}$ and constructs paths forwards or backwards by appending the node with maximal flow. Specifically, in order to identify inequalities (2.58) we construct paths from node s forwards, whereas for inequalities (2.59) we construct paths to node s backwards. To search for violations of inequalities (2.58) we then check whether each path originating from s contains the dropoff of a user $n + j$ but not its pickup j. Similarly, to search for violations of inequalities (2.59) we check whether each path terminating at s contains the pickup of a user j but not its dropoff $n+j$. The search is interrupted when one of the following conditions is met: (1) a violation of inequalities (2.58)-(2.59) is identified; or (2) the forward or backward path reaches a charging facility, a dummy origin, or destination depot without violating inequalities (2.58) and (2.59).

2.6 Numerical Experiments

In this section, we present numerical experiments designed to examine the performance of the proposed e-ADARP formulations and branch-and-cut algorithm. All programs are implemented in Julia 0.7.0 using the JuMP modeling language (2017) and the MILP solver Gurobi 7.0.1 on a 3.60 GHz Intel(R) Core(TM) with 16 Gb of RAM.

2.6.1 Test Instances

Two sets of instances are tested in the experiment: (1) Instances obtained by supplementing DARP benchmark instances from Cordeau (2006) with e-ADARP-related parameters, and (2) Instances based on ride-sharing data from Uber Technologies Inc. in 2011.

The instances proposed by Cordeau (2006) for the standard DARP are supplemented with charging stations, battery capacities, initial SOC requirements, final SOC requirements, recharge rates, and discharge rates. A single origin and destination depot is used for all vehicles, as in the original instances. That is, vehicles are limited to a specific destination depot and cannot choose from a set of potential destination depots. The capacity of all vehicles is set to three passengers and the maximum ride time of all users is set to 30 minutes. Recharge and discharge rates are given by the autonomous-electric shuttles manufacturer Navya and are set to represent a battery autonomy of 5 hours from a nominal capacity of 16.5 kWh. Namely, the recharge and discharge rates are both set to 0.055 kWh per minute (see `https://www.hevs.ch/media/document/1/fiche-technique-navettes-autonomes.pdf`). For the purpose of this experiment, all vehicles are assumed to start with full battery capacity, which is discharged on each route segment proportionally to its travel time. The effective battery capacity and initial battery charge are both set to 14.85 kWh. That is, vehicles are required to keep 10% of their nominal capacity at all times. Note that, with respect to the average travel time exhibited in these instances, each vehicle can visit at most approximately 20 nodes with a full effective battery capacity.

In the proposed analysis, the following convention is used for the instance names: <a/u><number of vehicles>-<number of customers>, where “a" and “u" are used for the instances adapted from Cordeau (2006) and from Uber respectively. The characteristics of the first instances are summarized in Table 2.4(a). The first column presents the instance name, followed by three columns presenting the time horizon T_p, number of customers n, and the number of vehicles $|\mathcal{K}|$. The last four columns present the number of constraints and variables (before pre-processing) in

Table 2.4: Benchmark Instances: (a) Instances adapted from Cordeau (2006); (b) Instances from Uber ride-shares in San Francisco, CA (USA).

				e-ADARP3		e-ADARP2	
Name	T_p **[min]**	**n**	$\|\mathcal{K}\|$	**# Cons**	**# Vars**	**# Cons**	**# Vars**
a2-16	480	16	2	19746	2990	9904	1786
a2-20	600	20	2	29486	4354	14550	2526
a2-24	720	24	2	41015	5974	20047	3394
a3-18	360	18	3	34447	5451	12891	2329
a3-24	480	24	3	57174	8949	21081	3631
a3-30	600	30	3	87923	13311	31571	5221
a3-36	720	36	3	124084	18537	44087	7099
a4-16	240	16	4	35866	5964	11245	2140
a4-24	360	24	4	74887	11924	22172	3876
a4-32	480	32	4	127995	19932	36829	6124
a4-40	600	40	4	194035	29988	54874	8884
a4-48	720	48	4	271841	42092	76050	12156
a5-40	480	40	5	235439	37475	56028	9265
a5-50	600	50	5	359418	56785	84664	13515

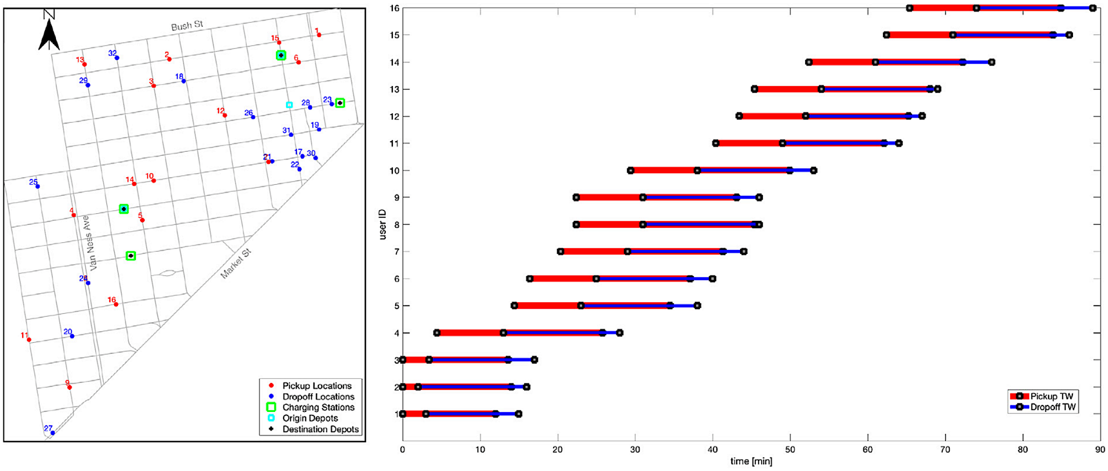
(a)

				e-ADARP3		e-ADARP2	
Name	T_p **[min]**	**n**	$\|\mathcal{K}\|$	**# Cons**	**# Vars**	**# Cons**	**# Vars**
u2-16	127	16	2	63296	3990	11348	2233
u2-20	166	20	2	36513	5546	16216	3053
u2-24	199	24	2	50094	7358	22324	4001
u3-18	161	18	3	43668	7095	13924	2731
u3-24	188	24	3	69947	11025	22291	4129
u3-30	237	30	3	104008	15819	33235	5815
u3-36	280	36	3	142511	21477	45619	7789
u4-16	93	16	4	47788	7964	12013	2427
u4-24	439	24	4	92195	14692	23132	4259
u4-32	183	32	4	150075	23468	37699	6603
u4-40	471	40	4	222327	34292	55945	9459
u4-48	272	48	4	310059	47164	78097	12827
u5-40	211	40	5	275315	42855	57151	9655
u5-50	284	50	5	409514	63365	84991	13985

(b)

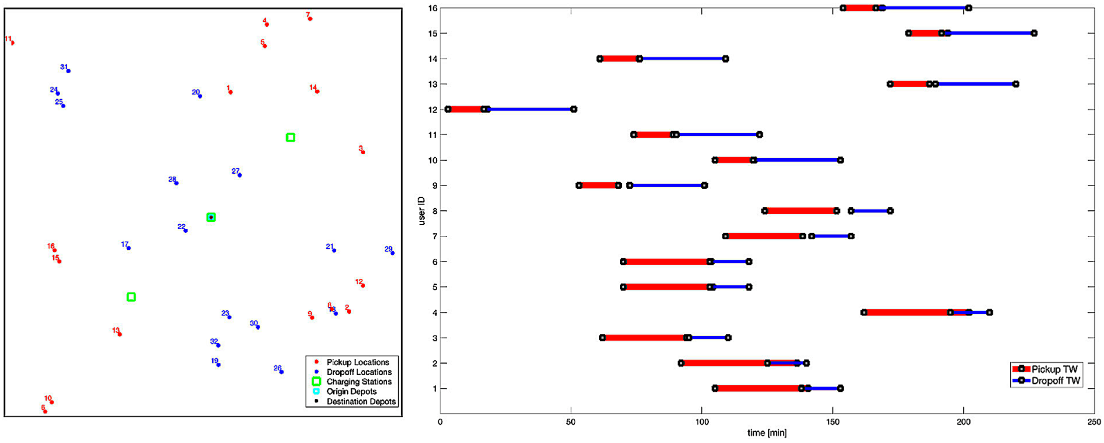
the resulting 3-index (e-ADARP3) and 2-index (e-ADARP2) MILP formulations. All of the presented instances are also replicated for three final minimum battery level ratios r. In particular, we analyze the case in which all vehicles need to return to one of the optional destination depots with at least 10%, 40% and 70% battery ratio levels (i.e. r=0.1, r=0.4, r=0.7).

The second set of instances is produced by extrapolating origin/destination locations and times from ride-sharing GPS logs in the city of San Francisco (CA, USA). The dataset, shared by Uber Technologies Inc. in 2011, contains a total of 1.2 million GPS logs, registered every 4 seconds from active Uber cars during one week. The dataset is processed prior to the analysis, removing invalid records, and extracting the pickup and dropoff locations for the remaining trips. The processed dataset contains about 25,000 Uber trips made during a one-week period. The daily number of trip requests varies from 2,000 to 6,000. The raw dataset is obtained from the GitHub repository at `https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata`. This study focuses on the ride-sharing requests that have been served in the Downtown/Civic Center districts.



(a) (b)

Figure 2.5: Instance a4-16: (a) Pickup and dropoff locations; (b) Pickups and dropoffs time windows after tightening. Note that the pickup location of any user with pickup ID i is denoted by i+16



(a) (b)

Figure 2.6: Instance u4-16: (a) Pickup and dropoff locations; (b) Pickups and dropoffs time windows after tightening. Note that the pickup location of any user with pickup ID i is denoted by i+16

Neighborhood information is obtained through the San Francisco Enterprise GIS Program (SFGIS) on the SF OpenData website. The requests from the weekend are aggregated into one single day, by neglecting the day of the request. Instances of different size are then generated by randomly selecting users at a maximum rate of one request every 3 minutes.

The San Francisco transportation network is extracted from OpenStreetMap (OSM). Information on electric vehicle charging station locations is obtained through the Alternative Fueling Station Locator from the Alternative Fuels Data Center (AFDC) of the U.S. Department of Energy. Dijkstra's shortest path algorithm is used to compute travel times between the pickup/dropoff locations, charging stations, and origin/destination depots. The travel times are estimated without considering traffic congestion, instead assuming that vehicles travel at a constant speed of 35 km/h. In the envisioned application AVs can provide non-stop ride-sharing services. As such, origin and destination depots are assumed to coincide with the set of potential charging stations. The depots also serve as temporary parking locations to provide ride-sharing services in a sequential time-horizon framework. In these instances, each vehicle may visit a maximum of five charging stations (randomly selected from the AFDC) and consequently may choose between 5 potential destination depots. Figure 2.5-2.6 show the pickup and dropoff locations, as well as their time windows, after tightening, for instances a4-16 and u4-16 respectively (i.e. 4 vehicles, 16 customers). Note that the instance shown in Figure 2.6 is substantially different from the one shown in Figure 2.5. In particular, all instances adapted from Cordeau, 2006 and from Uber, differ as follows: (1) The pickup and dropoffs in Figure 2.5(a) can appear anywhere, whereas in Figure 2.6(a) they can only appear on the transportation network; (2) after pre-processing, the time windows in Figure 2.5(b) are tighter than the time windows in Figure 2.6(b); (3) there is more overlap between the pickup and dropoff time windows of the same user in Figure 2.6(b) than in Figure 2.5(b) (i.e. dropoff time windows are more constraining in the Uber instances); and (4) there is more overlap between the time windows of different users in Figure 2.6(b) than in Figure 2.5(b) (i.e. users arrival rate is higher for the Uber instances).

In order to scale-up the study region we have focused on, travel times are multiplied by a factor of 2, battery consumptions are increased by 30%, and effective battery capacities are decreased to 3.5 kWh. Given this transformation, each vehicle can visit at most approximately 15 nodes, with a full effective battery capacity. Again, it is assumed that all vehicles depart from their origin depots with full battery capacities. The maximum ride times of all users are set to eight minutes and 15-minute time windows are applied at the dropoff locations. In addition, the planning horizon is obtained by considering the dropoff location with the latest service start time and adding the travel time to the furthest charging station. The characteristics of the second set of instances are summarized in Table 2.4(b). The table follows the same format as Table 2.4(a).

The objective function weight factors are set to $\{0.75, 0.25\}$, i.e. the total vehicle travel time accounts for 75% of the objective function score and the total excess ride time accounts for 25%. The two datasets used for the computational experiments are available on the website `https://people.epfl.ch/cgi-bin/people?id=255972&op=publications&lang=en&cvlang=en`.

2.6.2 Branch-and-Cut Results

Next, the performance of the proposed branch-and-cut framework is analyzed for e-ADARP3 and e-ADARP2. In both cases, the default branching strategy of Gurobi is used. In addition, cut generation procedures and separation heuristics are activated at every node in the search tree. Table 2.5 and Table 2.6 present the computational results obtained for the instances adapted from Cordeau (2006)

Table 2.5: Instances adapted from Cordeau (2006): Comparison between e-ADARP3 and e-ADARP2

	e-ADARP3					e-ADARP2				
Name	**CPU**	**Obj.**	**LB**	**Nodes**	**Cuts**	**CPU**	**Obj.**	**LB**	**Nodes**	**Cuts**
r=0.1										
a2-16	0.12	237.38*	237.38	170	121	0.02	237.38*	237.38	1	55
a2-20	6.67	279.08*	279.08	10866	752	0.07	279.08*	279.08	1	108
a2-24	2.58	346.21*	346.21	1768	1106	0.15	346.21*	346.21	132	131
a3-18	7.37	236.82*	236.82	10339	1466	0.08	236.82*	236.82	1	96
a3-24	21.23	274.81*	274.80	11022	1874	0.23	274.81*	274.81	299	182
a3-30	120.00	413.27	376.15	27196	2388	1.70	413.27*	413.27	2362	119
a3-36	120.00	481.72	458.43	20643	2940	1.78	481.17*	481.17	1561	206
a4-16	52.65	222.49*	222.49	48594	1826	0.06	222.49*	222.49	126	465
a4-24	120.00	310.84	275.47	22260	2057	0.52	310.84*	310.84	856	268
a4-32	120.00	413.02	308.14	11327	3585	10.20	393.96*	393.96	10389	164
a4-40	120.00	NA	371.71	6463	3658	8.62	453.84*	453.84	5128	386
a4-48	120.00	NA	375.72	2372	4502	120.00	554.54	526.96	36558	469
a5-40	120.00	490.49	279.56	2024	5242	19.03	414.51*	414.51	10270	666
a5-50	120.00	NA	359.50	1392	5191	120.00	559.17	531.15	28368	852
r=0.4										
a2-16	0.13	237.38*	237.38	165	106	0.03	237.38*	237.38	103	135
a2-20	5.97	280.70*	280.70	10277	730	0.83	280.70*	280.70	1663	89
a2-24	4.20	348.04*	348.04	3644	893	0.42	348.04*	348.04	786	105
a3-18	4.76	236.82*	236.82	5844	1178	0.07	236.82*	236.82	82	92
a3-24	32.89	274.80*	274.80	19257	1829	0.28	274.81*	274.81	279	186
a3-30	120.00	413.80	369.89	28757	1986	1.65	413.37*	413.37	2305	96
a3-36	120.00	489.99	452.33	20961	1741	5.11	484.14*	484.14	3855	204
a4-16	53.20	222.49*	222.49	47028	1400	0.09	222.49*	222.49	309	529
a4-24	120.00	311.03	281.23	25986	2467	0.66	311.03*	311.03	1170	278
a4-32	120.00	394.32	307.27	9190	4158	11.36	394.26*	394.26	11735	156
a4-40	120.00	NA	372.53	5215	712	6.96	453.84*	453.84	3622	430
a4-48	120.00	NA	381.69	3136	4247	120.00	554.60	529.22	38467	418
a5-40	120.00	454.81	281.44	1639	5811	20.35	414.51*	414.51	12282	727
a5-50	120.00	NA	372.81	1556	6258	120.00	560.50	528.91	26189	772
r=0.7										
a2-16	0.49	240.66*	240.66	1223	314	0.09	240.66*	240.66	1360	218
a2-20	120.00	NA	286.06	171592	529	120.00	NA	287.17	306306	103
a2-24	58.99	358.21*	358.21	75993	825	16.02	358.21*	358.21	37261	123
a3-18	10.71	240.58*	240.58	15927	1017	0.80	240.58*	240.58	2450	330
a3-24	49.29	277.72*	277.72	29723	1412	2.54	277.72*	277.72	4265	190
a3-30	120.00	NA	358.79	38105	2132	120.00	NA	417.06	134851	98
a3-36	120.00	NA	433.38	20902	2645	120.00	494.04	485.91	107714	193
a4-16	36.32	223.13*	223.13	33176	2246	1.12	223.13*	223.13	5996	690
a4-24	120.00	321.03	279.85	23100	2435	30.58	318.21*	318.19	73110	268
a4-32	120.00	NA	302.28	11142	3083	120.00	430.07	387.99	108524	131
a4-40	120.00	NA	372.75	5611	389	120.00	NA	443.62	62465	384
a4-48	120.00	NA	381.98	2107	5281	120.00	NA	524.92	35901	485
a5-40	120.00	NA	280.02	2264	5817	120.00	447.63	405.99	62627	582
a5-50	120.00	NA	357.51	1421	7008	120.00	NA	522.37	31725	751

and Uber respectively, with a time limit of 120 minutes. Note that in the following results, each charging stations may be visited at most once (i.e. the proposed charging-station nodes are not replicated). The first column in Tables 2.5-2.6 presents the instance name, the second to sixth columns present, for e-ADARP3, the CPU time in minutes, the best solution found, the lower bound, the number of explored nodes, and the number of generated cuts. The seventh to the last columns present the same information for e-ADARP2. Cases where the algorithm is unable to find a feasible solution within the given time limit are denoted by

Table 2.6: Instances from Uber: Comparison between the e-ADARP3 model and the e-ADARP2 model

	e-ADARP3					e-ADARP2				
Name	**CPU**	**Obj.**	**LB**	**Nodes**	**Cuts**	**CPU**	**Obj.**	**LB**	**Nodes**	**Cuts**
r=0.1										
u2-16	4.47	57.61*	57.61	8058	1025	0.35	57.61*	57.61	1251	1105
u2-20	0.32	55.59*	55.59	372	640	0.16	55.59*	55.59	1	679
u2-24	120.00	91.27	87.52	85942	1190	7.20	91.27*	91.27	13482	2081
u3-18	10.50	50.74*	50.74	9286	1251	0.18	50.74*	50.74	357	947
u3-24	120.00	68.78	60.95	32696	1717	2.17	67.56*	67.56	3819	2450
u3-30	120.00	76.99	70.44	20711	2021	7.30	76.75*	76.75	7399	4455
u3-36	120.00	109.12	93.01	14386	2186	18.08	104.04*	104.04	11540	3556
u4-16	111.94	53.58*	53.58	76603	1576	0.80	53.58*	53.58	2607	2113
u4-24	57.94	89.83*	89.83	22915	1607	0.22	89.83*	89.83	251	692
u4-32	120.00	NA	83.93	11342	2200	19.31	99.29*	99.29	11686	5093
u4-40	120.00	136.93	111.33	4546	3282	3.09	133.11*	133.11	1915	1093
u4-48	120.00	NA	110.50	2096	3902	120.00	148.30	134.48	19952	7497
u5-40	120.00	NA	94.21	1831	3624	120.00	121.86	114.12	25388	8259
u5-50	120.00	NA	114.06	1456	4920	120.00	143.10	132.69	20580	9141
r=0.4										
u2-16	3.83	57.65*	57.65	7477	1149	0.43	57.65*	57.65	1414	1263
u2-20	1.39	56.34*	56.34	2162	927	0.20	56.34*	56.34	219	445
u2-24	120.00	NA	85.00	85334	1252	12.62	91.63*	91.63	22501	1831
u3-18	12.88	50.74*	50.74	14101	1153	0.23	50.74*	50.74	451	1112
u3-24	120.00	67.77	61.20	34906	1673	3.68	67.56*	67.56	6589	3051
u3-30	120.00	78.15	70.13	20729	1756	5.61	76.75*	76.75	5016	4090
u3-36	120.00	NA	93.15	13127	1902	33.50	104.06*	104.06	19705	3710
u4-16	88.29	53.58*	53.58	67819	1341	0.74	53.58*	53.58	2374	2208
u4-24	85.55	89.83*	89.83	37381	2264	0.47	89.83*	89.83	383	836
u4-32	120.00	NA	84.41	10821	2399	44.46	99.29*	99.29	29909	5262
u4-40	120.00	NA	109.74	4001	2909	44.22	133.91*	133.91	31550	1098
u4-48	120.00	NA	110.43	2125	3833	120.00	NA	133.86	21713	8262
u5-40	120.00	NA	93.86	1799	4233	120.00	122.23	112.58	27739	8218
u5-50	120.00	NA	114.00	1475	4683	120.00	143.14	134.09	17899	9339
r=0.7										
u2-16	25.76	59.19*	59.19	63296	1117	5.64	59.19*	59.19	19883	1274
u2-20	2.23	56.86*	56.86	2609	827	1.20	56.86*	56.86	1999	1389
u2-24	120.00	NA	85.98	90525	1283	120.00	NA	90.83	151765	1652
u3-18	8.02	50.99*	50.99	7578	1163	0.40	50.99*	50.99	994	1374
u3-24	120.00	69.30	60.36	35613	1531	6.67	68.39*	68.39	10163	2080
u3-30	120.00	80.35	69.44	21115	1783	56.69	78.14*	78.14	58457	2908
u3-36	120.00	NA	92.30	14169	1917	120.00	105.79	104.37	73041	3526
u4-16	120.00	53.87	51.85	52303	1353	1.48	53.87*	53.87	4872	3304
u4-24	100.76	89.96*	89.96	41435	1581	0.38	89.96*	89.96	451	656
u4-32	120.00	NA	83.86	9265	2568	47.12	99.50*	99.50	30099	5173
u4-40	120.00	NA	109.53	4452	2424	120.00	NA	133.01	65051	826
u4-48	120.00	NA	111.64	1853	3862	120.00	NA	132.49	20208	7919
u5-40	120.00	NA	94.63	1711	4073	120.00	NA	109.28	29812	8070
u5-50	120.00	NA	113.94	1426	3546	120.00	144.36	133.33	18964	9177

Not Available (NA).

Comparing the computational results of the proposed branch-and-cut algorithm for e-ADARP3 and e-ADARP2 (shown in Table 2.5 and Table 2.6 respectively), reveals the following observations: (1) e-ADARP2 leads to considerably shorter CPU times with respect to e-ADARP3; (2) in the cases where the optimal solution has not been reached, e-ADARP2 finds stronger lower and upper bounds than e-ADARP3; (3) e-ADARP2 finds a feasible solution to more instances than e-ADARP3; (4) instances with up to 5 vehicles and 40 customers can be solved

Table 2.7: Instances from Uber: Root node lower bounds as a percentage of the upper bound

Instance	**LP3**	**LP2**	**GOC**	**SEC**	**SCC**	**TC**	**FC**	**RC**	**MRT**	**BM**	**CSW**	**Full**	**U.Bound**
r=0.1													
u2-16	81.68	84.87	92.38	89.81	91.40	90.85	88.08	92.91	51.14	86.27	91.05	92.95	57.61
u2-20	94.13	96.24	97.09	73.58	95.85	96.08	96.16	96.01	97.26	96.21	69.49	100.00	55.59
u2-24	72.14	80.64	82.57	82.03	81.29	81.84	60.63	82.84	66.01	55.83	81.10	87.28	91.27
u3-18	85.13	94.12	93.49	93.91	93.59	93.74	94.19	94.07	94.01	93.87	91.12	96.15	50.74
u3-24	78.50	88.05	88.20	88.60	77.78	88.62	88.45	87.84	89.51	88.34	71.62	91.22	67.56
u3-30	80.03	88.22	89.94	90.16	67.38	67.45	88.97	90.29	90.31	89.47	88.55	91.89	76.75
u3-36	80.91	87.44	60.01	65.07	59.97	61.19	66.20	88.46	89.35	88.68	66.78	90.02	104.04
u4-16	73.30	82.07	83.32	83.01	83.10	82.33	82.31	83.18	84.56	83.10	82.13	85.43	53.58
u4-24	76.54	72.82	87.41	88.31	71.98	69.99	73.65	87.10	96.63	87.86	49.32	85.66	89.83
u4-32	76.54	57.12	58.09	81.67	53.35	80.39	76.55	54.97	85.94	58.20	75.24	88.66	99.29
u4-40	74.77	94.49	53.61	76.27	64.85	57.16	54.79	63.63	97.38	74.14	57.68	97.09	133.11
u4-48	64.93	46.80	47.39	48.85	45.08	50.68	51.02	48.40	57.76	44.93	43.87	61.74	148.30
u5-40	69.64	49.01	48.68	48.27	47.23	51.30	49.17	49.98	53.87	48.09	49.10	54.02	121.86
u5-50	69.14	53.38	53.87	54.91	50.51	54.82	57.30	55.43	58.36	53.10	53.97	65.53	143.10
Avg.	76.96	76.80	74.00	76.03	70.24	73.32	73.39	76.79	79.43	74.86	69.36	84.83	
r=0.4													
u2-16	81.75	88.98	90.07	90.97	89.94	90.28	88.98	93.23	92.59	86.87	91.34	93.01	57.65
u2-20	92.11	93.93	95.84	95.65	94.65	95.27	94.86	95.38	93.78	94.65	94.69	97.92	56.34
u2-24	72.05	81.32	81.61	80.35	81.31	81.33	82.19	84.16	84.68	81.26	80.02	86.24	91.63
u3-18	84.91	91.09	93.55	93.59	94.18	93.69	94.12	93.68	94.30	93.67	93.71	94.08	50.74
u3-24	78.31	88.45	88.52	72.37	88.42	89.04	88.49	89.49	89.46	69.74	88.55	90.83	67.56
u3-30	80.19	89.00	90.40	71.72	75.79	89.71	88.79	89.77	88.73	66.56	64.41	91.09	76.75
u3-36	80.95	62.62	56.30	60.87	66.29	67.05	58.95	67.44	69.99	61.59	67.07	80.14	104.06
u4-16	44.69	48.83	41.98	49.27	49.57	48.91	49.37	49.20	50.39	49.57	48.98	51.22	89.83
u4-24	77.86	47.84	88.37	86.93	47.84	86.40	74.02	87.18	96.63	47.84	86.91	94.27	89.83
u4-32	76.40	57.88	75.92	76.61	53.25	58.07	58.87	75.53	77.45	75.61	74.08	88.56	99.29
u4-40	74.12	73.91	72.61	78.02	60.92	74.83	72.20	74.96	95.76	55.45	58.78	97.33	133.91
u4-48	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
u5-40	70.20	46.95	47.88	50.26	46.82	49.95	47.28	48.03	51.46	48.78	47.78	55.89	122.23
u5-50	69.10	52.54	52.67	54.62	49.26	55.28	55.65	55.28	61.38	53.26	53.71	65.01	143.14
Avg.	75.59	71.03	75.06	73.94	69.10	75.37	73.37	77.18	80.51	68.07	73.08	83.51	
r=0.7													
u2-16	78.41	88.41	49.49	89.75	88.42	88.02	88.41	88.21	90.11	87.38	87.98	90.71	59.19
u2-20	90.11	93.40	94.99	94.60	94.38	94.03	92.77	95.11	93.54	92.75	92.77	96.67	56.86
u2-24	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
u3-18	84.61	93.01	92.76	93.44	93.09	93.40	92.90	93.45	93.93	93.13	93.30	94.24	50.99
u3-24	77.70	86.94	86.93	88.02	77.92	87.26	88.14	87.73	88.69	87.26	87.10	89.82	68.39
u3-30	77.86	87.62	68.62	88.57	62.35	65.01	88.25	61.76	71.94	64.33	88.32	89.25	78.14
u3-36	79.36	61.96	60.40	64.66	58.27	58.23	59.09	67.70	87.41	59.16	58.11	88.22	105.79
u4-16	74.54	81.98	83.01	83.43	82.16	81.98	82.04	83.02	84.13	82.16	82.60	85.58	53.87
u4-24	76.78	65.67	88.23	86.16	47.77	71.52	71.68	86.75	96.48	47.77	86.78	93.79	89.96
u4-32	76.39	75.60	62.18	83.62	71.98	79.80	84.98	84.28	53.83	53.80	76.93	86.69	99.50
u4-40	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
u4-48	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
u5-40	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
u5-50	69.54	53.87	52.54	53.75	51.77	54.75	55.30	54.98	61.21	52.58	53.94	63.68	144.36
Avg.	78.53	78.85	73.92	82.60	72.81	77.40	80.36	80.30	82.13	72.03	80.78	87.86	

in Table 2.5; and (5) instances with up to 4 vehicles and 40 customers can be solved in Table 2.6.

In order to evaluate the strength of the proposed valid inequalities for the Uber instances, e-ADARP2 is supplemented with each inequality and calculate its performance at the root node. The results of such analysis are reported in Table 2.7, in which the root node lower bounds are reported as a percentage of the upper bound values. Column LP3 in Table 2.7 indicates the percentages obtained by solving the LP relaxation of e-ADARP3, whereas column LP2 represents the lower bounds obtained by solving the LP relaxation of e-ADARP2. The next eight columns present the bounds obtained when supplementing the LP relaxation of the 2-index model with valid inequalities from one of the following families: Generalize Order Constraints (GOC), Subtour Elimination Constraints (SEC), Strengthened Capacity Constraints (SCC), Tournament Constraints (TC), Fork Constraints (FC), Reachability Constraints (RC), Maximum ride time Constraints (MRT), battery-management Constraints (BM), or Charging Stations Walls Constraints (CSW). Column "Full" reports the bounds obtained when searching for all of the aforementioned families of valid inequalities. The last column presents the upper bound values which are either the optimal values if a solution is found within the two hour time limit or the best bounds within this time. Note that LP2 includes the separation of capacity, precedence, and pairing constraints. It is therefore possible that LP2 (with no inequalities) presents higher lower bounds than LP2 supplemented with one of the proposed inequalities. Nevertheless, as shown by column "Full", the interaction between all of the proposed inequalities always strengthens the lower bounds at the root node. The results show that, on average over all instances, Maximum ride time (MRT), Reachability (RC) and Subtour Elimination (SEC) constraints have the largest impact on average. However, the relative performance of each inequality is dependent on the minimal battery level ratio, certain inequalities might outperform others. In particular, when the minimal battery level requirement is most restrictive (i.e. $r = 0.7$), Charging Station Walls (CSW) constraints are among the most effective inequalities.

2.6.3 Sensitivity Analysis

This Section provides managerial insights highlighting the consequences of allowing multiple charging visits per station, employing an electric or a conventional internal-combustion fleet, considering operational and level-of-service oriented objectives.

Multiple charging visits can be modeled by replicating the charging station nodes and imposing symmetry breaking constraints (2.62). In Table 2.8 the effect of such replications on the solution quality and performance is analyzed for the Uber instances. The first column of Table 2.8 displays the instance name. The second to the sixth column report the CPU time (in minutes), objective value, optimality gap, number of visited stations while vehicles are en-route n_{S_R}, and number of visited stations before returning to the depot n_{S_D}, when only a single visit per charging station is allowed. The next five columns present the equivalent information for two allowed visits per charging stations. Finally, the last five columns present the equivalent information for three allowed visits per charging stations. The following observations can be drawn from the results in Table 2.8: (1) allowing multiple charging visits per station helps in finding a feasible integer solution when no feasible solution could be found with only one visit per station (e.g. instance u4-48 r=0.4); (2) allowing multiple charging visits per station might slightly improve the solution quality, where a maximum objective decrease by 1.72% is experienced by instance u2-16 r=0.7; (3) on average, providing 2 visits per charging station does

Table 2.8: Instances from Uber: Solution quality and performance when increasing the maximum number of charging visits per station.

	1 visit per station					2 visits per station					3 visits per station				
Name	**CPU**	**Obj.**	**Gap[%]**	n_{S_R}	n_{S_D}	**CPU**	**Obj.**	**Gap[%]**	n_{S_R}	n_{S_D}	**CPU**	**Objective**	**Gap[%]**	n_{S_R}	n_{S_D}
r=0.1															
u2-16	0.35	57.61	0.00	0	1	0.54	57.61	0.00	0	1	2.96	57.61	0.00	0	1
u2-20	0.16	55.59	0.00	0	1	0.11	55.59	0.00	0	1	0.43	55.59	0.00	0	1
u2-24	7.20	91.27	0.00	2	1	14.85	91.27	0.00	2	1	36.05	91.27	0.00	2	1
u3-18	0.18	50.74	0.00	0	0	0.40	50.74	0.00	0	0	0.85	50.74	0.00	0	0
u3-24	2.17	67.56	0.00	0	0	3.94	67.56	0.00	0	0	3.24	67.56	0.00	0	0
u3-30	7.30	76.75	0.00	0	0	5.01	76.75	0.00	0	0	5.24	76.75	0.00	0	0
u3-36	18.08	104.04	0.00	0	3	17.82	104.04	0.00	0	3	18.81	104.04	0.00	0	3
u4-16	0.80	53.58	0.00	0	0	0.92	53.58	0.00	0	0	0.93	53.58	0.00	0	0
u4-24	0.22	89.83	0.00	0	1	0.27	89.83	0.00	0	1	0.32	89.83	0.00	0	1
u4-32	19.31	99.29	0.00	0	0	52.23	99.29	0.00	0	0	52.11	99.29	0.00	0	0
u4-40	3.09	133.11	0.00	0	2	3.70	133.11	0.00	0	2	2.35	133.11	0.00	0	2
u4-48	120.00	148.30	9.32	0	4	120.00	148.37	10.12	0	4	120.00	149.14	9.71	0	4
u5-40	120.00	121.86	6.35	0	1	120.00	121.86	10.14	0	2	120.00	121.86	7.38	0	1
u5-50	120.00	143.10	7.27	0	0	120.00	142.83	5.42	0	1	120.00	142.83	6.96	0	1
r=0.4															
u2-16	0.43	57.65	0.00	0	2	0.86	57.65	0.00	0	2	3.02	57.65	0.00	0	2
u2-20	0.20	56.34	0.00	0	2	0.99	56.34	0.00	0	2	4.31	56.34	0.00	0	2
u2-24	12.62	91.63	0.00	3	1	13.36	91.27	0.00	2	2	54.28	91.27	0.00	2	2
u3-18	0.23	50.74	0.00	0	1	0.38	50.74	0.00	0	1	1.13	50.74	0.00	0	1
u3-24	3.68	67.56	0.00	0	2	2.11	67.56	0.00	0	2	3.63	67.56	0.00	0	2
u3-30	5.61	76.75	0.00	0	2	5.82	76.75	0.00	0	2	3.31	76.75	0.00	0	2
u3-36	33.50	104.06	0.00	0	3	18.80	104.06	0.00	0	3	21.66	104.06	0.00	0	3
u4-16	0.74	53.58	0.00	0	0	1.29	53.58	0.00	0	0	1.91	53.58	0.00	0	0
u4-24	0.74	89.83	0.00	0	2	0.40	89.83	0.00	0	2	0.49	89.83	0.00	0	2
u4-32	44.46	99.29	0.00	0	2	33.86	99.29	0.00	0	2	30.37	99.29	0.00	0	2
u4-40	44.22	133.91	0.00	0	4	104.93	133.68	0.00	1	4	120.00	134.01	0.67	1	4
u4-48	120.00	NA	NA	NA	NA	120.00	150.96	12.00	1	4	120.00	150.78	11.70	1	4
u5-40	120.00	122.23	7.89	0	4	120.00	122.22	8.48	0	4	120.00	121.96	6.27	0	4
u5-50	120.00	143.14	6.32	0	4	120.00	142.83	5.69	0	4	120.00	143.48	8.13	0	4
r=0.7															
u2-16	5.64	59.19	0.00	2	1	2.76	58.17	0.00	1	2	21.91	58.17	0.00	1	2
u2-20	1.20	56.86	0.00	1	2	2.90	56.86	0.00	1	2	14.58	56.86	0.00	1	2
u2-24	120.00	NA	NA	NA	NA	120.00	97.50	7.23	3	2	120.00	NA	NA	NA	NA
u3-18	0.40	50.99	0.00	0	3	0.68	50.99	0.00	0	3	2.54	50.99	0.00	0	3
u3-24	6.67	68.39	0.00	0	3	4.33	68.06	0.00	1	3	8.44	68.06	0.00	1	3
u3-30	56.69	78.14	0.00	1	3	120.00	78.16	0.97	1	3	120.00	78.16	1.31	1	3
u3-36	120.00	105.79	1.34	1	3	120.00	107.65	4.17	1	2	120.00	106.18	2.42	1	3
u4-16	1.48	53.87	0.00	0	3	2.13	53.87	0.00	0	3	2.81	53.87	0.00	0	3
u4-24	0.38	89.96	0.00	1	2	0.49	89.83	0.00	1	2	1.32	89.83	0.00	1	2
u4-32	47.12	99.50	0.00	0	4	33.85	99.50	0.00	0	4	76.03	99.50	0.00	0	4
u4-40	120.00	NA	NA	NA	NA	120.00	137.49	3.16	2	3	120.00	137.61	3.27	2	4
u4-48	120.00	NA	NA	NA	NA	120.00	NA	NA	NA	NA	120.00	NA	NA	NA	NA
u5-40	120.00	NA	NA	NA	NA	120.00	125.14	14.11	0	5	120.00	124.18	10.48	1	5
u5-50	120.00	144.36	7.64	0	5	120.00	164.16	18.84	2	5	120.00	144.10	7.14	1	5

not significantly increase CPU times, with an average increase by 1.21 minutes; (4) the increase in CPU time might vary substantially between different instances, with a maximum increase by 63.31 minutes (instance u3-30 r=0.7) and a maximum decrease by 14.70 minutes (instance u3-36 r=0.4); (5) two visits per station is the upper bound on the number of charging visits per station for the instances which converged to the optimal solution; and (6) providing more than two visits per station might increase the optimality gap when the optimal solution has not been found by the time limit.

Table 2.9 presents solution details for the Uber instances and provide a comparison when employing internal combustion vehicles. Instances which converge to the optimal solution within the time limit are selected, when two charging

Table 2.9: Instances from Uber, Electric fleet compared to an internal combustion (IC) fleet: Number of utilized vehicles (Vehs), Total vehicle travel time cost (TT), Users excess ride time cost (ERT), Total number of visited charging stations en-route (n_{S_R}), Total charging time en-route (E_{S_R}), Total number of visited charging stations off-route (n_{S_D}), Total charging time off-route (E_{S_D})

	Electric fleet – 2 visits per station								IC fleet			
Name	**Vehs**	**Obj.**	**TT**	**ERT**	n_{S_R}	E_{S_R}	n_{S_D}	E_{S_D}	**Vehs**	**Obj**	**TT**	**ERT**
r=0.1												
u2-16	2	57.61	76.81	0.00	0	0.00	1	11.12	2	57.61	76.81	0.00
u2-20	2	55.59	73.70	1.24	0	0.00	1	6.06	2	55.59	73.70	1.24
u2-24	2	91.27	116.98	14.14	2	34.78	1	6.32	2	89.66	114.83	14.14
u3-18	3	50.74	67.65	0.00	0	0.00	0	0.00	3	50.74	67.65	0.00
u3-24	3	67.56	86.08	12.01	0	0.00	0	0.00	3	67.56	86.08	12.01
u3-30	3	76.75	100.83	4.50	0	0.00	0	0.00	3	76.75	100.83	4.50
u3-36	3	104.04	133.79	14.80	0	0.00	3	10.10	3	103.50	131.48	19.57
u4-16	3	53.58	68.76	8.06	0	0.00	0	0.00	3	53.58	68.76	8.06
u4-24	3	89.83	118.21	4.67	0	0.00	1	9.96	3	89.83	118.21	4.67
u4-32	4	99.29	129.19	9.59	0	0.00	0	0.00	4	99.29	129.19	9.59
u4-40	4	133.11	168.40	27.25	0	0.00	2	0.92	4	133.11	168.40	27.25
r=0.4												
u2-16	2	57.65	76.86	0.00	0	0.00	2	23.56	2	57.61	76.81	0.00
u2-20	2	56.34	74.70	1.24	0	0.00	2	39.08	2	55.59	73.70	1.24
u2-24	2	91.27	116.98	14.14	2	34.75	2	56.92	2	89.66	114.83	14.14
u3-18	3	50.74	67.65	0.00	0	0.00	1	11.27	3	50.74	67.65	0.00
u3-24	3	67.56	86.08	12.01	0	0.00	2	34.25	3	67.56	86.08	12.01
u3-30	3	76.75	100.83	4.50	0	0.00	2	50.11	3	76.75	100.83	4.50
u3-36	3	104.06	132.22	19.57	0	0.00	3	86.53	3	103.50	131.48	19.57
u4-16	3	53.58	68.76	8.06	0	0.00	0	0.00	3	53.58	68.76	8.06
u4-24	3	89.83	118.21	4.67	0	0.00	2	25.79	3	89.83	118.21	4.67
u4-32	4	99.29	129.19	9.59	0	0.00	2	123.08	4	99.29	129.19	9.59
u4-40	4	133.68	169.06	27.55	1	1.27	4	59.71	4	133.11	168.40	27.25
r=0.7												
u2-16	2	58.18	76.45	3.36	1	31.69	2	30.12	2	57.61	76.81	0.00
u2-20	2	56.86	75.40	1.24	1	30.50	2	39.08	2	55.59	73.70	1.24
u2-24	2	97.50	125.13	14.62	3	65.28	2	59.21	2	89.66	114.83	14.14
u3-18	3	50.99	67.99	0.00	0	0.00	3	31.11	3	50.74	67.65	0.00
u3-24	3	68.06	85.43	15.96	1	7.03	3	65.86	3	67.56	86.08	12.01
u3-30	3	78.16	102.71	4.50	1	11.75	3	65.09	3	76.75	100.83	4.50
u3-36	3	107.65	136.99	19.63	1	58.93	2	62.46	3	103.50	131.48	19.57
u4-16	3	53.87	69.14	8.06	0	0.00	3	33.91	3	53.59	68.76	8.06
u4-24	3	89.83	118.21	4.67	1	63.71	2	73.32	3	89.83	118.21	4.67
u4-32	4	99.50	128.58	12.26	0	0.00	4	114.20	4	99.29	129.19	9.59

visits per station are allowed. Unlike electric vehicles, internal combustion vehicles can operate during the whole planning horizon without the need to refuel. The second to ninth column in Table 2.9 present results for an electric fleet in which 2 visits per charging station are allowed, as follows: the number of utilized vehicles, the objective function value, the total vehicle travel times (in minutes), the total user excess times (in minutes), the total number of visited charging stations while the vehicles are en-route, the total charging time at the visited charging stations while vehicles are en-route (in minutes), the total number of visited stations before returning to the selected destination depots (i.e. off-route), and the total charging time before returning to the depot (in minutes). The last three columns present the number of utilized vehicles, the objective function value, total vehicle travel

Table 2.10: Instances from Cordeau (2006): Comparison between the e-ADARP2 model, the e-ADARP2 model when employing an internal combusion fleet, the e-ADARP2 model when only considering total vehicle travel time in the objective function, the standard DARP results.

	e-ADARP2			e-ADARP2 IC fleet			e-ADARP2 1obj.			DARP		
Name	**TT**	**ERT**	**Gap[%]**	**TT**	**ERT**	**Gap[%]**	**TT**	**ERT**	**Gap[%]**	**TT**	**ERT**	**Gap[%]**
r=0.1												
a2-16	299.26	51.73	0.00	299.26	51.73	0.00	294.25	72.98	0.00	294.25	73.00	0.00
a2-20	347.89	72.63	0.00	347.89	72.63	0.00	344.83	112.55	0.00	344.83	112.55	0.00
a2-24	433.26	85.08	0.00	433.26	85.08	0.00	413.12	157.41	0.00	413.12	157.41	0.00
a3-18	304.34	34.25	0.00	304.34	34.25	0.00	300.48	119.06	0.00	300.48	119.06	0.00
a3-24	355.78	31.87	0.00	355.78	31.87	0.00	344.83	120.31	0.00	344.83	120.31	0.00
a3-30	500.69	150.99	0.00	500.69	150.99	0.00	494.85	197.18	0.00	494.85	197.18	0.00
a3-36	600.85	122.12	0.00	600.85	122.12	0.00	583.19	244.37	0.00	583.19	244.37	0.00
a4-16	286.45	30.60	0.00	286.45	30.60	0.00	282.68	44.49	0.00	282.68	44.49	0.00
a4-24	388.83	76.88	0.00	388.83	76.88	0.00	375.02	159.65	0.00	375.02	159.65	0.00
a4-32	502.34	68.80	0.00	502.34	68.80	0.00	485.50	165.20	0.00	485.50	165.20	0.00
a4-40	562.44	128.04	0.00	562.44	128.04	0.00	557.69	167.64	0.00	557.69	167.64	0.00
a5-40	524.75	83.78	0.00	524.75	83.78	0.00	498.41	296.37	0.00	498.41	296.37	0.00
r=0.4												
a2-16	299.26	51.73	0.00	299.26	51.73	0.00	294.25	72.98	0.00	294.25	73.00	0.00
a2-20	350.06	72.63	0.00	347.89	72.63	0.00	347.73	112.55	0.00	344.83	112.55	0.00
a2-24	435.70	85.08	0.00	433.26	85.08	0.00	433.46	157.41	0.00	413.12	157.41	0.00
a3-18	304.34	34.25	0.00	304.34	34.25	0.00	300.48	119.06	0.00	300.48	119.06	0.00
a3-24	355.78	31.87	0.00	355.78	31.87	0.00	344.83	120.31	0.00	344.83	120.31	0.00
a3-30	500.83	150.99	0.00	500.69	150.99	0.00	494.84	197.18	0.00	494.85	197.18	0.00
a3-36	615.10	91.29	0.00	600.85	122.12	0.00	588.12	232.74	0.00	583.19	244.37	0.00
a4-16	286.45	30.60	0.00	286.45	30.60	0.00	282.68	44.49	0.00	282.68	44.49	0.00
a4-24	394.48	60.68	0.00	388.83	76.88	0.00	375.02	159.65	0.00	375.02	159.65	0.00
a4-32	502.74	68.80	0.00	502.34	68.80	0.00	485.90	165.20	0.00	485.50	165.20	0.00
a4-40	562.44	128.04	0.00	562.44	128.04	0.00	557.69	167.64	0.00	557.69	167.64	0.00
a5-40	524.75	83.78	0.00	524.75	83.78	0.00	498.41	296.37	0.00	498.41	296.37	0.00
r=0.7												
a2-16	303.64	51.73	0.00	299.26	51.73	0.00	298.63	73.00	0.00	294.25	73.00	0.00
a2-20	NA	NA	NA	347.89	72.63	0.00	NA	NA	NA	344.83	112.55	0.00
a2-24	445.80	95.44	0.00	433.26	85.08	0.00	444.63	194.74	0.00	413.12	157.41	0.00
a3-18	309.36	34.25	0.00	304.34	34.25	0.00	303.71	115.54	0.00	300.48	119.06	0.00
a3-24	359.67	31.87	0.00	355.78	31.87	0.00	350.69	126.34	0.00	344.83	120.31	0.00
a3-30	NA	NA	NA	500.69	150.99	0.00	NA	NA	NA	494.85	197.18	0.00
a3-36	618.01	122.12	1.65	600.85	122.12	0.00	599.75	261.41	0.00	583.19	244.37	0.00
a4-16	282.68	44.49	0.00	286.45	30.60	0.00	282.68	44.49	0.00	282.68	44.49	0.00
a4-24	388.43	107.55	0.00	388.83	76.88	0.00	381.70	149.72	0.00	375.02	159.65	0.00
a4-32	545.17	84.80	9.79	502.34	68.80	0.00	491.23	165.20	0.79	485.50	165.20	0.00
a4-40	NA	NA	NA	562.44	128.04	0.00	NA	NA	NA	557.69	167.64	0.00
a5-40	568.97	83.61	9.30	524.75	83.78	0.00	528.21	283.80	6.97	498.41	296.37	0.00

time and user excess ride time when adopting an internal combustion fleet. The following observations can be drawn from the results in Table 2.9: (1) in some of the proposed instances, vehicles need to recharge en-route and at the end of the planning horizon (e.g. instance u2-24); (2) as expected, en-route recharge can be avoided when the final battery requirements are less restrictive (i.e. r=0.1); (3) vehicles recharge longer when the minimal battery level requirement at the selected destination depots is more restrictive (i.e. r=0.7); (4) 3 vehicles out of 4 are utilized to serve the customers in instances u4-16 and u4-24; and (5) utilizing an electric fleet might incur in higher objective function values due to deviations towards charging stations, which may result in either (or both) an increase in the operational cost or a decrease in the quality of service provided to the users.

The e-ADARP extends the standard DARP through several features which have an effect on the total vehicle travel time and user excess ride time. Table 2.10 analyses the effect derived from two such features, i.e. the choice of the objective function and the use of the electric fleet. The analysis provides results for instances from Cordeau (2006) with up to 5 vehicles and 40 customers. The second to the fourth column report the total vehicle travel time, user excess ride time and optimality gaps for the original e-ADARP2 model, in which up to 1 visit per charging station is allowed. The successive columns present the same results for three relaxed versions of e-ADARP2, i.e. “e-ADARP2 IC fleet", “e-ADARP2 1 obj.", and “DARP". The first omits battery considerations. The second omits excess ride time from the objective function. The third omits both. Note that models “e-ADARP2 1 obj." and “DARP" provide optimal routing and scheduling solutions which exclusively minimize the total vehicle travel time. Consequently, the scheduling solutions might be sub-optimal in terms of the total user excess ride time. In order to compare the total excess-ride time values provided by all of the proposed models in Table 8, the optimal routing decisions in “e-ADARP2 1 obj." and “DARP" are fixed and the schedules are post-optimized to minimize the total excess time, as a secondary objective. The post optimization is performed by solving a linear program with the objective of minimizing the total user excess time and the constraints remaining from the original problem after fixing the routing decisions. The resulting total user excess time is then reported under columns “ERT" for models “e-ADARP2 1 obj." and “DARP". Given that the “e-ADARP2 IC fleet" and “DARP" models both assume the use of non-electric vehicles, the results are not affected by changes in the minimum battery level ratio, r.

The following observations are extracted from the results in Table 2.10: (1) requiring vehicles to return with at least 10% of their battery inventory produces no battery range anxiety; (2) employing an internal combustion fleet saves on average about 2% of the total routing cost, while excess ride time experiences variations by up to 30%; (3) finding feasible solutions to instances $a2-20$, $a3-30$, and $a4-40$ ($r=0.7$) is challenging due to battery considerations; (4) with respect to e-ADARP2, “e-ADARP2 1 obj." produces solutions in which the total vehicle travel time is only decreased by 2.6%, while the total user excess time is sensibly increased by about 124.45%; and (5) similar to (4), with respect to “e-ADARP2 1 obj.", “DARP" produces solutions in which the total vehicle travel time is only decreased by 2.48%, while the total user excess ride time is increased by about 117.89%. The results in (4) and (5) demonstrate that considering operational and quality-related criteria in a weighted-sum objective drastically decreases user inconvenience for a moderate increase in transportation cost.

2.7 Summary

In this study, we introduce the electric autonomous dial-a-ride problem (e-ADARP), which generalizes the standard dial-a-ride problem for the case where electric autonomous vehicles are employed. We present two mixed-integer linear program formulations, namely a 3-index model and a 2-index model. New features incorporated in the problem definition include charging stations, multiple destination depots, partial recharge times, and final battery level requirements. We employ a weighted objective function that accounts for the total travel time of all vehicles and excess ride time of all users. We develop both new valid inequalities and lifted inequalities from literature by taking into consideration specific properties of the e-ADARP. In addition, we design purpose-based separation heuristics to separate the proposed inequalities in a branch-and-cut framework.

Computational experiments are carried out on selected benchmark instances from DARP literature adapted for the e-ADARP, and on a new set of instances based on Uber ride-sharing requests in the city of San Francisco. Results show that, when used in a branch-and-cut framework, the 2-index model outperforms the 3-index model for all of the tested datasets. Based on the root-node analysis, among all of the designed valid inequalities maximum ride time and reachability constraints are the most effective for the instances derived from Uber ride-shares. Nevertheless, when battery-management aspects are most important, charging station walls constraints are among the most effective inequalities. A sensitivity analysis shows that, for the tested instances, the number of optional visits to charging stations might influence algorithmic performance and solution quality. In particular, CPU times are mostly impacted when the number of optional charging visits per station exceeds the maximum number needed to improve the solution quality. Finally, a sensitivity analysis shows that operating an electric fleet may result in increased operational cost or decreased quality of service (or both) with respect to a traditional internal combustion fleet, due to detours to charging stations.

3

A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic Electric Autonomous Dial-a-Ride Problem

This chapter is based on the article:

- C. Bongiovanni, M. Kaspi, J.-F. Cordeau, and N. Geroliminis (2020). "A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic Electric Autonomous Dial-a-Ride Problem". Working paper

3.1 Introduction

This chapter models and solves the dynamic electric autonomous dial-a-ride problem, whose main differences with respect to the standard DARP are introduced in Section 1.3. The problem is solved through a two-phase metaheuristic within an event-based simulation framework. In particular, a first insertion phase is applied to myopically assign each request arrival to a vehicle (e.g. Coslovich, Pesenti, and Ukovich, 2006; Diana and Dessouky, 2004). The assignment can be performed greedily, in consideration of the total cost provided by the objective function. A second re-optimization phase is applied to iteratively revisit decisions taken in the first phase through intra- and inter- vehicle route exchanges. Such exchanges may be performed in light of forecasted demands (e.g. Peleda et al., 2019), however the most common practice remains to solve sequences of static, myopic, trip-assignment sub-problems (Powell et al., 2002). The re-optimization phase can be triggered after the first phase or at a pre-defined frequency. For example, it may occur after a number of myopic insertions, the denial of a new upcoming request, and after selected statistics from the vehicle plans meet given thresholds.

Ride-sharing demand exhibits repeated patterns during the days of the week. As such, local search-based metaheuristics solve similar sub-problems sharing multiple similarities in terms of demand distribution and route plans. Assuming that similar neighborhood moves generate comparable effects for similar instances and routing solutions, information acquired from previously solved sub-problems can become useful in finding new ways to guide the search during on-line operations.

In this work, the information is constructed off-line by exhaustively examining the neighborhood defined by each operator before moving to the next iteration in the search. After the retrieval of sufficient historical data, we extract statistical models relating selected problem features to the performance of several competing algorithms. The statistical models are learned off-line and are efficiently re-applied to take decisions on-line. The machine learning-based metaheuristic is denoted by ML-LNS and, to the best of our knowledge, it is proposed for the first time in this work. At each iteration during the search, the competing algorithms destroy and repair vehicle routes in light of different considerations. The learned statistical models are used to draw the destroy-repair couple according to their predicted performance at each iteration during the search. We measure performance by means of the expected objective function improvement. Worsening and intermediate infeasible solutions are also considered and reflected into the expected objective function improvement.

In ML-LNS, the prediction task replaces the roulette wheel mechanism proposed in the adaptive large neighborhood search (ALNS) metaheuristic (Ropke and Pisinger, 2006) and is based on a large dataset containing examples of moves from all of the competing LNS algorithms on several instances and routing solutions. Note that, the proposed optimization framework fundamentally differs from ALNS. Specifically, ALNS assumes that the future of the search should be driven by the success of the competing operators on previous iterations, while ML-LNS assumes that the future of the search should be driven by the estimated algorithm performances at the current iteration. In this work, the prediction task is tackled through ensemble learning, namely random forest (RF) regression (Breiman, 2001). Random forests have a similar performance compared to other popular machine learning methodologies, such as support vector machines, while having numerous practical advantages. What makes RF suitable for this work is its ability to automatically select features, provide a variable importance measure, handle outliers and very big datasets. Note that several methodologies may be employed and compared for the prediction task, even within the machine learning field. However, a thorough comparison between prediction methodologies does not lie within the scope of this study.

In the context of on-line settings, there are a number of advantages to employing a machine learning mechanism rather than the adaptive mechanism, as in ALNS. First, given that the learning phase is performed off-line, a machine learning approach does not need a warm-up time to adjust the relative scores through several iterations. Instead, the scores are directly provided for each sub-problem during the on-line LNS search. Second, a machine learning approach exploits impactful similarities between new encountered sub-problems and the several past static sub-problems used during the off-line training phase. As such, a machine learning approach efficiently directs the search of new problems towards directions that have been proven statistically successful. Third, the predictions derived from a machine learning approach allow to efficiently select destroy-repair operators concurrently rather than separately, a practice that is generally avoided in ALNS due to computational burden (Ropke and Pisinger, 2006). Finally, this work devises a holistic solution approach, which can be employed for both static and dynamic problems, and identifies impactful problem-related features, which can be equally useful for solving other vehicle routing problems, especially variants of the pickup and delivery problem (PDP) and DARP.

To summarize, this work proposes five main contributions: (1) we introduce a new and relevant variant of the dynamic DARP, i.e. the dynamic e-ADARP; (2) we represent realistic dynamic operations through an event-based simulation approach; (3) we propose a novel two-phase metaheuristic employing a machine learning approach within a local-search based metaheuristic; (4) we identify vehicle routing problem features that impact local search algorithms and can be utilized to effectively guide search mechanisms; (5) we produce a massive labeled dataset used for training purposes. Numerical experiments are performed on real ride-sharing data from Uber Technologies Inc. The ML-LNS is benchmarked against state-of-the-art approaches, namely LNS, ALNS, and a random selection approach. Furthermore, an analysis of the results provide insights on the vehicle routing features which are most determinant when taking decisions about the re-optimization.

The rest of this chapter is organized as follows: Section 3.2 formally defines the dynamic e-ADARP, Section 3.3 describes the solution approach, Section 3.4 focuses on the ML approximation, Section 3.5 describes the labeling and feature selection for the dynamic e-ADARP, Section 3.7 contains the results of the numerical experiments. A summary and future directions are provided in Section 3.8.

3.2 Problem Definition

In the dynamic e-ADARP, a fleet of e-AVs $\mathcal{K} = \{1, \ldots, k\}$, leaving from origin depots $o^k \in \mathcal{O}$, are to give service to a set of unknown transportation requests within a fixed planning horizon H. Without loss of generality, assume that initial vehicle routes are non-empty and vehicles are planned to give service to n initial transportation requests, characterized by given pickup locations $\mathcal{P} = \{1, \ldots, n\}$, dropoff locations $\mathcal{D} = \{n+1, \ldots, 2n\}$ and time windows $[arr_i,\ dep_i]$ around the desired pickup times or dropoff times. We denote by $\mathcal{N}$ the set containing both pickup and dropoff locations. Each transportation request $i \in \{1, \ldots, n\}$ is represented by a load l_i (i.e. $l_i \geq 0$ for $i \in \mathcal{P}$ and $l_i = -l_{i-n}$ for $i \in \mathcal{D}$), a non-null service time d_i, and a maximum ride time u_i. All requests are to be served within the plannig horizon H, after which vehicles return to one of the optional destination depots $f^k \in \mathcal{F}$. All nodes are connected by arcs with travel time t_{ij}, hence each vehicle chooses the destination depot f^k which is closest to the last visited location i, i.e. $f^k = \operatorname{argmin}_{j \in \mathcal{F}} t_{ij}$.

Vehicles are heterogeneous in terms of their loading capacity C^k, homogeneous in terms of their battery capacity Q, and characterized by initial battery levels B_0^k corresponding to their current state-of-charge (SOC). Vehicle batteries can be recharged at uncapacitated charging stations $\mathcal{S}$. It is assumed that empty vehicles can visit multiple charging stations along a route, can partially recharge, and return with minimal battery levels r at destination depots $\mathcal{F}$. The amount of energy recharged at the charging stations in $\mathcal{S}$ is proportional to the time E_s^k spent at the facilities. In particular, the amount of energy transferred per unit time is controlled by the charging rate α_s (e.g. fast charging, slow charging). Similarly, battery consumptions $\beta_{i,j}$, with i and $j \in \mathcal{V} = \mathcal{O} \cup \mathcal{N} \cup \mathcal{S} \cup \mathcal{F}$, can be inferred from the travel times t_{ij} combined with other features by means of an energy consumption model (e.g. Pelletier et al., 2017b; Goeke and Schneider, 2015).

Initial vehicle plans are constructed by finding minimum cost routes and observing time window, capacity, maximum ride time, and battery constraints. Five types of decisions variables are used to make decisions on the vehicle plans. A binary decision variable x_{ij}^k denotes whether vehicle k sequentially visits locations i and $j \in \mathcal{V}$. T_i^k represents the time at which vehicle k begins service at location $i \in \mathcal{V}$, L_i^k represents its load after service, and B_i^k represents its battery state at the beginning of service. In addition, E_s^k denotes the recharge time of vehicle k at stations $s \in \mathcal{S}$. R_i represents the excess ride time of user $i \in \mathcal{P}$, that is the extra time users spend on board of the vehicles with respect to a taxi service. As in the work presented in Chapter 2, the cost function consists of a weighted-sum objective comprising the total vehicle travel time and user excess ride time.

At time $h \leq H$, in the course of operations, the transportation system receives a new transportation request with pickup location $\tilde{p}$ and dropoff location $\tilde{d}$. If more than one new transportation request appear at the same time, the requests are treated independently and in the order of arrival. No information regarding the request location is available beforehand. However, its booking time may be issued a few minutes in advance with respect to the desired pickup time. Depending on the current states and future tasks assigned to the vehicles, the new request $\{\tilde{p}, \tilde{d}\}$ may be either accepted or rejected. Service is only rejected if the new request cannot be feasibly inserted into the existing vehicle routes, in consideration of the constraints imposed by the e-ADARP problem. Denying service to the new request incurs into a fixed-cost penalty c_3 which affects the objective function through an auxiliary binary decision variable $z_{\tilde{p}}$. The cost penalty c_3 may be computed as a proportion of the objective function value. As such, existing vehicle plans may need to be revisited in order to find the optimal trade-off between the following components of the objective function: (1) feasibly insert the new transportation request, (2) decrease operational cost, (3) and increase the level of service. This is obtained by re-optimizing a current static sub-problem at time h, which is composed of all information related to the current vehicle locations $\bar{o}^k$, the current vehicle plans, and the new transportation request.

3.2.1 Static Sub-Problems in the Dynamic e-ADARP

Static sub-problems are essentially a variant of the static e-ADARP (presented in Chapter 2) with some adaptations considering the dynamic environment. For example, in the dynamic e-ADARP new transportation requests may be rejected, requests which were previously accepted need to be served, vehicles are on the move, and pickup/dropoff services happened in the past cannot be modified. As a result, vehicle may be non-empty, part of the excess ride time characterizing planned requests may have been used, and vehicle charging phases may need to be interrupted or modified. Next, we present a mathematical formulation of the resulting static sub-problem encountered at each re-optimization phase.

At the arrival of the new request, each vehicle $k \in \mathcal{K}$ may be en-route or have just arrived to a location $\bar{j}^k$ consisting of a pickup in $\mathcal{P}$, a dropoff in $\mathcal{D}$, a charging station in $\mathcal{S}$, or a destination depot in $\mathcal{F}$. As such, vehicle travel times from the current vehicle locations $\bar{o}^k$ to $\bar{j}^k$, i.e. $t_{\bar{o}^k,\bar{j}^k}$, are computed to account for the remaining portion of the route to be traveled. Direct deviation times $t_{\bar{o}^k,\tilde{p}}$ from the vehicle current locations to the pickup of the new request are also simply computed by employing the law of cosines (Weisstein, 2020). Service at the pickup or droppoff nodes cannot be preempted, however vehicles that are waiting at pickupor dropoff nodes can be redirected to other locations. Note that parking facilities may not be available at locations $j \in \mathcal{N}$. Hence vehicle waiting times are bounded from above (e.g. by a few minutes) through a maximal waiting time w_j. In contrast, recharging phases at stations $s \in \mathcal{S}$ may be preempted if the vehicle charge level is sufficient to allow traveling between locations.

At the arrival of the new requests $\{\tilde{p}, \tilde{d}\}$, vehicles may be non-empty. In particular, some requests with pickups $\hat{\mathcal{P}}^k \subseteq P$ and dropoffs $\hat{\mathcal{D}}^k \subseteq \mathcal{D}$ may have been picked-up by vehicle $k \in \hat{\mathcal{K}} \subseteq \mathcal{K}$ without being dropped-off. In this case, the initial occupancy of all of such vehicles is to be reduced by the number of requests currently on board. From a modeling perspective, dummy pickup nodes $\hat{\mathcal{P}}^k$ are generated at the corresponding vehicle locations and need to be served. In order to produce routing solutions starting with sequential visits all of the dummy pickups $\hat{\mathcal{P}}^k$, the following inequalities are enforced:

$$\sum_{i=1}^{|\hat{\mathcal{P}}^k|-1} x^k_{o^k,i} + x^k_{i,i+1} = |\hat{\mathcal{P}}^k| - 1 \quad \forall k \in \hat{\mathcal{K}}$$

Note that the order in which dummy pickup nodes $\hat{\mathcal{P}}^k$ are served is not relevant.

Furthermore, the maximum ride time and the direct travel time $t_{i,n+i}$ of each request in $i \in \hat{\mathcal{P}}^k$ are both reduced by the time already traveled on board of vehicle k, as follows:

$$u_i = u_i - (h - T^k_i) \quad \forall k \in \hat{\mathcal{K}}, i \in \hat{\mathcal{P}}^k$$

$$t_{i,n+i} = t_{i,n+i} - (h - T^k_i) \quad \forall k \in \hat{\mathcal{K}}, \forall i \in \hat{\mathcal{P}}^k$$

With such pre-processing steps, static e-ADARP sub-problems at time h can be formulated as the MILP presented in Table 3.1.

Table 3.1: 3-indexed formulation for static e-ADARP sub-problems at time h

$$\min c_1 \cdot \sum_{k\in\mathcal{K}} \sum_{i,j\in\mathcal{V}} t_{ij} x_{ij}^k + c_2 \cdot \sum_{i\in\mathcal{P}} R_i + c_3 \cdot \sum_{i\in\mathcal{P}\cup\tilde{p}} z_i \tag{3.1}$$

subject to:

$$\sum_{j\in\mathcal{P}\cup\mathcal{S}\cup\mathcal{F}\cup\{\tilde{p}\}} x_{o^k j}^k = 1 \qquad \forall k \in \mathcal{K} \tag{3.2}$$

$$\sum_{j\in\mathcal{F}} \sum_{i\in\mathcal{D}\cup\mathcal{S}\cup\{\tilde{d}\}} x_{ij}^k = 1 \qquad \forall k \in \mathcal{K} \tag{3.3}$$

$$\sum_{k\in\mathcal{K}} \sum_{\substack{j\in\mathcal{N}\cup\{\tilde{p},\tilde{d}\}\\ j\neq i}} x_{ij}^k = 1 \qquad \forall i \in \mathcal{P} \tag{3.4}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}\setminus\mathcal{F}} x_{i,\tilde{p}}^k \leq 1 \tag{3.5}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}\setminus\mathcal{F}} x_{ij}^k = 1 - z_j \qquad \forall j \in \mathcal{P}\cup\tilde{p} \tag{3.6}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in\{o^k\}\cup\mathcal{D}\cup\{\tilde{d}\}\cup\mathcal{S}} x_{ij}^k \leq 1 \qquad \forall j \in \mathcal{F}\cup\mathcal{S} \tag{3.7}$$

$$\sum_{\substack{j\in\mathcal{V}\cup\{\tilde{p},\tilde{d}\}\\ j\neq i}} x_{ij}^k - \sum_{\substack{j\in\mathcal{V}\cup\{\tilde{p},\tilde{d}\}\\ j\neq i}} x_{j,i}^k = 0 \qquad \forall k \in \mathcal{K}, i \in \mathcal{N}\cup\mathcal{S}\cup\{\tilde{p},\tilde{d}\} \tag{3.8}$$

$$\sum_{\substack{j\in\mathcal{N}\cup\{\tilde{p},\tilde{d}\}\\ j\neq i}} x_{ij}^k - \sum_{\substack{j\in\mathcal{N}\cup\{\tilde{p},\tilde{d}\}\\ j\neq n+i}} x_{j,n+i}^k = 0 \qquad \forall k \in \mathcal{K}, i \in \mathcal{P}\cup\{\tilde{p}\} \tag{3.9}$$

$$T_i^k + d_i + t_{i,n+i} \leq T_{n+i}^k \qquad \forall k \in \mathcal{K}, i \in \mathcal{P}\cup\{\tilde{p}\} \tag{3.10}$$

$$arr_i \leq T_i^k \leq dep_i \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\} \tag{3.11}$$

$$T_{n+i}^k - T_i^k - d_i \leq u_i \qquad \forall k \in \mathcal{K}, i \in \mathcal{P}\cup\{\tilde{p},\tilde{d}\} \tag{3.12}$$

$$T_i^k + t_{ij} + d_i - M_{i,j}^k(1 - x_{ij}^k) \leq T_j^k \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\}, j \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\}, i \neq j | M_{i,j}^k > 0 \tag{3.13}$$

$$x_{ij}^k \Rightarrow T_j^k - (T_i^k + t_{ij} + d_i) \leq w_j \qquad \forall k \in \mathcal{K}, i \in \mathcal{N}\cup\{\tilde{p},\tilde{d}\}, j \in \mathcal{N}\cup\{\tilde{p},\tilde{d}\}, i \neq j \tag{3.14}$$

$$R_i \geq T_{n+i}^k - T_i^k - d_i - t_{i,n+i} \qquad \forall k \in \mathcal{K}, i \in \mathcal{P}\cup\{\tilde{p}\} \tag{3.15}$$

$$L_i^k = 0 \qquad \forall k \in \mathcal{K}, i \in o^k \tag{3.16}$$

$$L_i^k \geq \max(0, l_i) \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}\cup\{\tilde{p},\tilde{d}\} \tag{3.17}$$

$$L_i^k \leq \min(C^k, C^k + l_i) \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}\cup\{\tilde{p},\tilde{d}\} \tag{3.18}$$

$$L_i^k + l_j - G_{i,j}^k(1 - x_{ij}^k) = L_j^k \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\}, j \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\}, i \neq j | G_{i,j}^k > 0 \tag{3.19}$$

$$L_i^k = 0 \qquad \forall k \in \mathcal{K}, i \in \mathcal{F}\cup\mathcal{S} \tag{3.20}$$

$$B_i^k = B_0^k \qquad \forall k \in \mathcal{K}, i \in o^k \tag{3.21}$$

$$B_j^k = B_i^k - \beta_{i,j} + Q(1 - x_{ij}^k) \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\} \setminus \mathcal{S}, j \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\} \setminus \{o^k\}, i \neq j | Q > 0 \tag{3.22}$$

$$B_j^k = B_s^k + \alpha_s E_s^k - \beta_{s,j} + Q(1 - x_{s,j}^k) \qquad \forall k \in \mathcal{K}, s \in \mathcal{S}, j \in \mathcal{P}\cup\{\tilde{p}\}\cup\mathcal{F}\cup\mathcal{S}, s \neq j \tag{3.23}$$

$$E_s^k = T_s^k - t_{i,s} - T_i^k + M_{i,s}^k(1 - x_{i,s}^k) \qquad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, i \in \mathcal{D}\cup\{\tilde{d}\}\cup\mathcal{S}\cup\{o^k\}, i \neq s | Mi, s^k > 0 \tag{3.24}$$

$$Q \geq B_s^k + \alpha_s E_s^k \qquad \forall k \in \mathcal{K}, s \in \mathcal{S} \tag{3.25}$$

$$B_i^k \geq rQ \qquad \forall k \in \mathcal{K}, i \in \mathcal{F} \tag{3.26}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\}, j \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\} \tag{3.27}$$

$$B_i^k \geq 0 \qquad \forall k \in \mathcal{K}, i \in \mathcal{V}\cup\{\tilde{p},\tilde{d}\} \tag{3.28}$$

$$E_s^k \geq 0 \qquad \forall k \in \mathcal{K}, s \in \mathcal{S} \tag{3.29}$$

The objective function (3.1) minimizes the weighted-sum objective composed of three components, namely the total traveled distance, the total excess ride time, and the penalty for denying the new request. Note that the objective function pushes towards solutions including the new request if weight factor c_3 is sensibly larger than weight factors c_1 and c_2. Constraints (3.2)-(3.3) ensure that all vehicles exit the origin depots and return to one of the available destination depots. Constraints (3.4) ensure that all of the requests in the vehicles plan are served. Constraints (3.5) allow vehicles to serve the new request $\{\tilde{p}\}$. If the new request is served, the auxiliary decision variable $z_{\tilde{p}}$ is set to zero and the third component of the objective no longer contributes to the total cost (3.6). Note that feasible solutions only consider solutions with $z_i = 0$ for $i \in P$. However, during the solution process, relaxations of the problem may consider intermediate infeasible solutions with $z_i > 0$ $\forall i \in \mathcal{P}$. Charging stations may be accessed from dropoff locations or other charging stations (3.7). Constraints (3.8) ensure flow conservation, while constraints (3.9) ensure that the same vehicle serves both the pickup and the dropoff of each request. Furthermore, pickup-dropoff precedences are enforced in constraints (3.10). Time window and maximum ride time constraints are set in (3.11)-(3.12). Service start times between nodes are computed through constraints (3.13) and bounded at pickup/dropoff locations by their maximal wait times (3.14). User excess ride times are set in (3.15). Vehicle loads are initialized through constraints (3.16). Loads are bounded by constraints (3.17)-(3.18) and are updated between consequent nodes according to (3.19). Charging stations and destination depots may be only accessed by empty vehicles (3.20). Constraints (3.21) set the initial battery levels while the SOC between following nodes is tracked through constraints (3.22). Vehicle battery inventories can be re-increased after visiting a charging facility (3.23), where they may recharge according to (3.24) and up to the vehicle maximum battery capacity (3.25). Vehicles have to return to the selected destination depots with some minimal battery charge (3.26). Finally, (3.27)-(3.29) are integrality constraints. Note that constraints (3.13), (3.19), (3.24) are linearized through time-dependent and load-dependent big-M parameters which can be computed as proposed in Chapter 2. Finally, the provided MILP formulation is a hard combinatorial problem which cannot be solved in nearly real-time through exact solution approaches, which is important for dynamic problems. Indeed, as shown in Chapter 2, small-sized problems of up to 4 vehicles and 24 customers can be solved through an exact solution approach in less than one minute.

3.3 Solution Approach

In order to solve the dynamic e-ADARP, we devise a two-phase metaheuristic approach which, in this work, is tested on simulated dynamic scenarios. In Section 3.3.1, we briefly present the event-based simulation framework designed for the e-ADARP. Then, in Section 3.3.2 we describe the two phase approach, followed by a focused description of the second machine learning-based re-optimization phase (3.3.3).

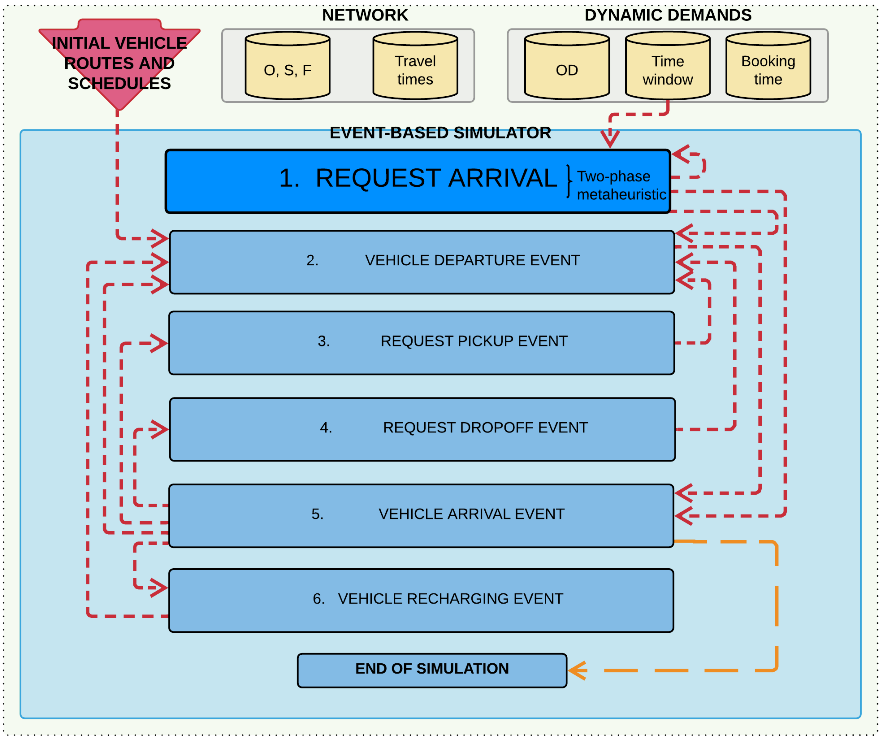


Figure 3.1: Flowchart for the event-based simulation

3.3.1 Event-Based Simulation

To represent realistic operations, the dynamic e-ADARP is simulated through an event-based approach, depicted by the flow chart in Figure 3.1. The input to the event-based simulation considers information on the transportation network (i.e. origin/destination depots, charging stations, travel times) and initial vehicle routes. Without loss of generality, it is assumed there are no pre-booked requests and that initial vehicle routes are only composed of vehicle origin depots, charging stations, and destination depots. Vehicle origin depots are chosen at random among the set of available origin depots in the transportation network. Vehicles are not allowed to wait at origin depots and are instead only allowed to idle at charging stations, before returning to one of the optional destination depots at the end of the planning horizon H. Each vehicle chooses the charging station and destination depot which minimizes travel-time distance. A set of n dynamic demands appear during the planning horizon H and are characterized by given pickup-dropoff locations, time windows, and stochastic booking times. For each request $i = \{1, \dots, n\}$, the booking time is obtained by generating an in-advance booking time which is subtracted from the earliest arrival time arr_i. In-advance booking times are computed as independent and identically distributed random variables from an exponential distribution with rate parameter λ, assumed to be homogeneous among requests. Larger rate parameters λ induce instantaneous booking times.

The generated input enters the simulation framework, which considers events related to vehicle and request occurrences. Specifically, vehicle events consider departure, arrival, and charging occurrences. Request events consider booking, pickup, and dropoff occurrences. The simulation is initialized by generating an event list composed of vehicle departure events from the origin depots and a first request arrival event. Vehicle departure events trigger arrival events at the following locations from the vehicle plans. Arrival events appear before eventual vehicle waiting phases, which are taken into account before producing recharge, pickup, or dropoff events. Vehicle departure events are triggered after service (i.e. pickup, dropoff, recharging). The arrival of each request is treated through a two-phase metaheuristic. If the request is accepted and feasibly inserted, the relative vehicle plan is updated. If the inserted request is to be served immediately, vehicle tasks may be modified, as well as the event list. There are two types of vehicle tasks that may be modified following an instantaneous insertion of a new request. Vehicles that are currently waiting or recharging may instantaneously depart to provide service to the newly inserted request. Vehicles that are currently en-route towards a specific location may instantaneously deviate towards the newly inserted request. In this second case, travel times are updated to account for the portion of the route which has already been traveled. Note that vehicles are not allowed to instantaneously depart and give service to new requests if they are providing service elsewhere. Furthemore instantaneous departures are only allowed in conformity with time windows, maximum ride time, and vehicle capacity considerations for all requests. Each request arrival event triggers a following request arrival event, which is selected in the order of the generated booking times from the input list of dynamic demands. The simulation terminates after the arrival of each vehicle at the chosen destination depots. With respect to the latter, note that the arrival of new requests may modify the destination depot choice for the selected vehicle.

In this work the stochasticity of the simulation is represented by the arrival of new requests. As such, the two-phase heuristic, and specifically the second re-optimization phase, is only triggered after the arrival of new requests. However, the proposed event-based simulation can be easily modified to trigger the second re-optimization phase in light of other changing conditions or optimization policies (e.g. traffic congestion, fixed-time-interval polishing, etc.) or to simultaneously treat multiple arrivals at once.

3.3.2 Two-phase Metaheuristic for Static e-ADARP Sub-Problems

This section describes the two-phase metaheuristic, which is triggered any time a request arrival event is detected, as highlighted in Figure 3.1. Consider an instance $i \in \{\text{instances}\}$, characterized by planned transportation requests and a new transportation request j. The current routing solution $s \in \{\text{solutions}\}$ has total operational cost $f(s)$, which is composed of the 3-term weighted objective function from equation 3.1, and uses a high penalty c_3 for each unassigned request j. As explained in Section 3.2, the goal of the two-phase metaheuristic is to find the right balance between the following components of the objective function: (1) feasibly insert the new transportation request, along with all other inserted requests; (2) decrease operational cost; (3) and increase the level of service. If j is feasibly inserted through a first greedy insertion phase, the goal of the second phase reduces to finding a trade-off between components (2) and (3) of the objective function. If component (1) is infeasible, then the new transportation request is rejected.

First Phase: Greedy Insertion Algorithm

The first phase is designed to select a vehicle that can feasibly accommodate each new transportation request j, received at time $h < H$. New transportation requests are represented by a generation time and a time window around the pickup or dropoff location. Note that the time window around the dropoff/pickup can be easily computed from its pickup/dropoff by considering the user maximum ride time u_i. Furthermore, time windows can be tightened by considering the vehicle current locations and tasks at time h. For example, a vehicle that is currently picking up a request i needs to first terminate its task before traveling from i to j. As a result, each new request features vehicle-specific earliest arrival times arr_j, which depend on the vehicle current locations and tasks. Instead, the latest departure time dep_j is vehicle-independent. Vehicles that cannot arrive to the pickup location of j by its latest arrival time dep_j are not candidates for the insertion.

Forward and backward slack times can be used to facilitate decisions about changes in the schedules (Savelsbergh, 1992). For the e-ADARP, the maximum time interval by which a specific user i and all preceding/following nodes in the vehicle schedule can be pushed backward/forward without violating time window constraints can be computed as in Section 3.2 in Diana and Dessouky (2004). Note that, in the case of the e-ADARP, service times include eventual recharging phases. Forward slack times F_i and backward slack times $\bar{F}_i$, computed as proposed in Parragh et al. (2009), are used to identify segments of the vehicle routes which may feasibly contain the pickup and the dropoff of j, separately. That is, having computed forward/backward slack times for all requests i within a vehicle plan, it is possible to identify segments of the plan where the insertion of the pickup or dropoff of j would not violate time window constraints. For example, the pickup of j certainly needs to be served after the last node i that satisfies the condition $T_i - \bar{F}_i < arr_j$. Similarly, the pickup of j certainly needs to be served before the first node i that satisfies the condition $T_i + F_i > dep_j$. In consideration of capacity constraints, such segments can be further restricted. That is, knowing the vehicle maximum capacity C and loads from the current static plan, the pickup of the new request j cannot be inserted when the vehicle is planned to travel at capacity.

The feasibility of a specific pickup-dropoff insertion option within a candidate vehicle can be further pre-processed with respect to time-window and battery feasibility. Time-window feasibility is linearly checked by assuming vehicles can start providing service at the first node in the vehicle plan at time h and by testing that time windows are not violated at any of the following nodes. Initial battery feasibility is assessed by assuming vehicles terminate current idling/recharging phases as soon as they have reached the minimal state of charge allowing to directly serve the pickup and dropoff of the new request, as well as all of their already planned requests. Note that we do not consider the insertion of new intermediate charging visits along the vehicle paths. This assumption is employed to limit the computational burden of the greedy insertion phase. However, a new recharging/idling visit is always appended at the end of each vehicle route by selecting the charging facility minimizing the travel time from its last visited location. Assuming that the vehicle deviation to serve the new transportation request requires a battery expense, it is sufficient to check that the subtraction of such expense from the vehicle initial battery states at the visited recharging facilities does not return negative values. If battery constraints are violated at any of the recharging facilities, the vehicle is assumed to be battery infeasible. Its limited battery inventory is consequently planned to be replenished at the new recharging visit, appended at the end of its route.

After pre-processing, vehicle schedules for given pickup-dropoff insertions can be constructed by employing the algorithm proposed in Bongiovanni, Kaspi, and Geroliminis (2020) and presented in Chapter 4. Successively, it is possible to check for maximum ride time constraints by identifying the users that would experience increased excess ride time, given the insertion of the new request j. For each candidate vehicle, multiple insertions of the new transportation request are evaluated against the objective function in equation 3.1. Finally, the chosen insertion is the one resulting in the lowest objective function value.

3.3.3 Second Phase: Machine Learning-Based Large Neighborhood Search

The second phase is triggered to revisit previous decisions taken in the first phase as to increase the level of service, described by the weighted objective function 3.1. To this aim, the ML-LNS destroys and repairs the current routing solution over multiple iterations $i \in \{1, \dots, k\}$. At each iteration, the search uses a specific destroy-repair couple $(d(i), r(i))$, which is selected among m^2 competing algorithms (d_l, r_n), with $l, n \in \{1, \dots, m\}$. Destroy-repair couples $(d(i), r(i))$, for $i \in \{1, \dots, k\}$, are selected according to the pseudo-code presented in Algorithm 1 and explained next.

At the beginning of iteration i, the destroy level q_i is drawn from a uniform bounded interval. The destroy level, instance, and routing solution information forms the input data vector $\mathbf{x}_i = \{x_{i1}, \dots, x_{id}\}$ which is fed to train destroy-repair regression models $\xi_{(d_l, r_n)}$, with $l, n \in \{1, \dots, m\}$. The regression models return the expected performance of each destroy-repair couple at the current iteration i, i.e. the expected objective improvement $y_{(d_l, r_n)}(i)$. Note that the expected objective improvement may be negative, since worsening solutions, either feasible or infeasible, are allowed to be explored during the search. As such, an exponential function is used to positively transform all values $y_{(d_l, r_n)}(i) \rightarrow exp(x \times y_{(d_l, r_n)}(i))$, while maintaining information on their relative magnitude. The exponential transform is smoothened by a factor x in order to avoid over-penalizing the choice of algorithms when one of them out-performs the others. Note that the use of the exponential function is also desirable for the proposed problem for the following features: 1) it highlights the difference between worsening and improving operators; 2) when

Algorithm 1: ML-LNS heuristic

```
Input: instance $i \in \{instances\}$, current solution $s \in \{solutions\}$, $j \in \{newrequest\}$, number of LNS
       iterations $k$, $m$ destroy models $d_l$, $m$ repair models $r_n$, regression models $\xi_{(d_l,r_n)}$ for
       $l, n \in \{1, \ldots, m\}$, smoothening factor $x$, LNS exit rule
Output: $s_{\text{best}}$
Initialize: $s_{\text{best}} = s$, $s_{\text{current}} = s$, $z_{\tilde{p}} = 0$ if $j = \emptyset$, $z_{\tilde{p}} = 1$ otherwise;
for $i \rightarrow$ 1:k do
    Draw: $q_i \in \mathbb{N}$;
    Compute: $\mathbf{x}_i$ from $i$, $s_{\text{current}}$, and $q_i$;
    Predict: $\xi_{(d_l,r_n)} : \mathbf{x}_i \rightarrow y_{(d_l,r_n)}(i)$ for $l, n \in \{1, \ldots, m\}$;
    Transform: $y_{(d_l,r_n)}(i) \rightarrow exp(x \times y_{d_l,r_n}(i))$ for $l, n \in \{1, \ldots, m\}$;
    Compute: $p_{d_l,r_n}(i) = \frac{y_{(d_l,r_n)}(i)}{\sum_{l=1}^{m} \sum_{n=1}^{m} y_{(d_l,r_n)}(i)}$ [%];
    Draw: $p_{d_l,r_n}(i) \rightarrow (d(i), r(i))$;
    Compute: $s^{'}, z_i, j = r(i)(d(i)(q_i \cup \{i|z_i = 1\}))$;
    if $f(s^{'}) < f(s_{best})$ & $\{i|z_i = 1\} \setminus \{j\} = \emptyset$ then
        $s_{\text{best}} = s^{'}$;
    if accept($f(s^{'}), f(s_{current})$) is true then
        $s_{\text{current}} = s^{'}$;
    if exit = true then
        break;
```

all algorithms perform similarly, it highlights the smallest difference between them. Successively, the predicted transformed performances are normalized and return the relative proportion of improvement per destroy-repair couple $p_{d_l,r_n}(i)$ at iteration i. The destroy-repair couple to be employed at the current LNS iteration, i.e. $(d(i), r(i))$, is drawn according to the predicted proportions $p_{d_l,r_n}(i)$. Operator $d(i)$ selects q_i requests to be removed from s. A request bank z_i is initialized to take into account the insertion of the new request j, if available, and updates the set of requests to be re-inserted to $q_i \cup \{i|z_i = 1\}$. The request bank may be updated after the re-insertion phase through $r(i)$, as explained next. Operator $r(i)$ repairs the solution through an iterative process, which may take up to $|q_i \cup \{i|z_i = 1\}|$ iterations. Each pending request is independently inserted into the current vehicle plans, according to $r(i)$. All feasible re-insertions belonging to different vehicles are selected to update the vehicle plans and the request bank. If more than one re-insertion belongs to the same vehicle, the minimum cost solution is selected. The remaining requests are re-inserted through $r(i)$ on the updated vehicle routes and up until the set of currently pending requests is emptied. During the iterative process, it is possible that some of the pending requests $q_i \cup \{i|z_i = 1\}$ cannot be feasibly re-inserted. These requests are placed into the request bank z_i, which penalizes the objective function and is fed at the following ML-LNS iteration. Note that during the re-insertion phase, the new request j may also be feasibly assigned to a vehicle route. In this case, j is updated to an empty set and the new request needs to be served as all the other planned requests in the vehicle routes. New incumbent solutions need to contain all of the removed requests, with the exception of the new request j if still available. That is, if $\{i|z_i = 1\} \setminus \{j\} \neq \emptyset$, the solution is infeasible. Finally, define $s^{'}$ as the new solution obtained by destroying the pending requests from s through $d(i)$ and repairing them through $r(i)$. If $s^{'}$ is feasible and

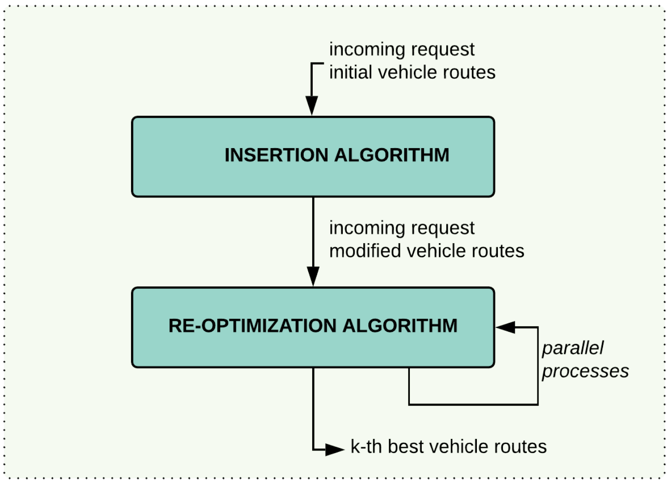


Figure 3.2: Considered re-optimization process

reduces total cost $f(s_{best})$, the incumbent solution s_{best} is updated. Note that a simple comparison between $f(s)$ and $f(s^{'})$ is not sufficient when the employed penalty cost is homogeneous between new and planned requests. For example, suppose the current solution is penalized by the existence of the new request j but it is feasible for all other requests. Successively, suppose that a neighborhood move $s^{'}$ has modified the vehicle routes such that j is inserted but another request $\bar{j}$ is placed in the bank. In this case, the solution $s^{'}$ is infeasible, although the objective function $f(s^{'})$ may be lower than $f(s)$ (i.e. if serving j comes at a lower cost than serving $\bar{j}$). The acceptance criterion, based on SA, allows the search to explore worsening solutions $s^{'}$ (including infeasible solutions with respect to a non-empty request bank) and always updates the current solution to $s^{'}$ when $f(s^{'}) < f(s)$. Instead, the acceptance criterion may reject worsening neighborhood moves at each iteration i and with iteration-dependent probabilities. The search, initially composed of k iterations, may be prematurely terminated, for example if the maximal number of sequential non-improving iterations or the maximum time limit (e.g. 5 seconds) is exceeded.

Finally, note that most commercial processors nowadays mount multiple cores (e.g. up to 8-12 cores) and that computations of the ML-LNS do not need more than a few cores (e.g. 2 cores). As such, in practice, the second phase can be run multiple times in parallel on each encountered sub-problem to be re-optimized, as shown in Figure 3.2. At the end of each parallel re-optimization, the best solution (or the k-th best) can be selected.

3.4 Prediction Problem

The prediction models employed within the ML-LNS are obtained through a statistical learning approach using a large dataset containing tuples $(\mathbf{x}_i, y_i)$. In this section, we define the prediction problem and depict the machine learning approximation.

3.4.1 Definition

Consider a problem instance i and an incumbent solution s characterized by aggregated features (i.e. input variables) $\{X_1, \ldots, X_d\}$. These features may correspond to scalar, boolean, or more generally categorical values (e.g. number of requests in the vehicle future plans, the presence of a new upcoming request, current vehicle tasks etc.). The d-dimensional vector $\mathbf{x}_i = \{x_{i1}, \ldots, x_{id}\}$ denotes one realization of such features and defines a datapoint. Tuples $(\mathbf{x}_i, y_i)$, with $i = \{1, \ldots, N\}$, are a collection of examples of problems for which the output y_i (e.g. the expected percentage improvement) is known, i.e. the labeled set. Sample an arbitrary large subset of examples within the labeled set, called the training set with size N_{train}. A supervised learning approach aims at estimating a function $\xi : \mathcal{X} \rightarrow \mathcal{Y}$ which best maps the input space to the output space through the examples provided by the training dataset. If $\mathcal{Y} = \mathbb{R}$, ξ is a regressor (e.g. it estimates the expected percentage improvement). The goal of the regression problem is to minimize a loss function $\mathcal{L}$ measuring the discrepancy between the predicted and known output values from the training dataset, with size N_{train}. The generalization capabilities of the resulting model is checked on a test dataset with size $N_{\text{test}} < N_{\text{train}}$. The test dataset consists of the examples that remain from the labeled set that are not in the training set. There exists multiple metrics providing the expected prediction error. In the proposed work, model performance is estimated by employing the following metrics: (1) Mean absolute error (MAE, also known as L-1 loss), (2) Root mean square error (RMSE, the root of the MSE or L-2 loss), (3) The coefficient of determination (R^2). The three measures are summarized here below:

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} |y_i - {y_i}^p|$$

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} (y_i - {y_i}^p)^2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N_{\text{test}}} (y_i - {y_i}^p)^2}{\sum_{i=1}^{N_{\text{test}}} (y_i - \bar{y}_i)^2}$$

Here, y_i denotes the observed data, ${y_i}^p$ the predicted data, and $\bar{y}_i$ the mean of the observed data. Note that the MAE and MSE can also be expressed with respect to the mean of the observed data (i.e. $rMAE$ and $rRMSE$) when the results are difficult to interpret. For a discussion exploring the meaning and differences between the proposed validation measures, the reader is referred to the on-line review in Grover (2018).

3.4.2 Machine Learning Approximation

In this work, the prediction problem is tackled through ensemble learning, namely random forest (RF) regression (Breiman, 2001). The goal of ensemble learning is to combine simple and fast learners to obtain better performance. In the case of RF, the weak learners are classification and regression trees (CART) (Breiman, 2017). CART recursively partition the input space through a series of binary splits chosen through a mathematical model which maximizes a goodness of split function. The depth of the tree is typically controlled by hyper-parameters which provide an upper bound on the maximal number of splits and, consequently, sub-regions. An indication of the values to be adopted for such hyper-parameters may be found in the literature (Liaw, Wiener, et al., 2002). Appropriate hyper-parameter values can be optimized for the treated problem through a search mechanism (e.g. grid search, random search) and k-fold cross validation.

For a regression problem, the value of each sub-region is decided by computing the average of the samples in the sub-region (the “average vote"). The predicted value for a new input point is then obtained by passing the point through the tree until a final node (or sub-region) is reached. In order to increase the precision of the prediction, multiple decision trees are typically combined through a technique called *bagging*, which essentially generates multiple models based on bootstrap samples of the input space (Breiman, 1996). Random forests attempt to further increase the performance of the predicted models and de-correlate trees by choosing a subset of the feature space at every split in the tree. The final prediction is the aggregation of the predictions of all models. For a regression task, the aggregation corresponds to the average of the predicted values. For random forests, as for any other ML methodology using bagging, the performance of the predicted models can be also measured by the out of bag (OOB) error, that is the average prediction error from each training datapoint using only the trees that did not contain it in their bootstrap sample.

Finally, random forests provide an intuitive way to interpret the average feature importance, measured by the average impurity decrease. Impurity is measured as the average squared difference between the labels of the samples in the current node and the current prediction at the node (i.e. the average of the samples in the node). Impurity decrease measures the reduction of impurity between successive nodes in the trees. Then, the feature importance within a tree is interpreted as the total impurity decrease brought by each feature within the tree. For a forest, the total impurity is averaged across all trees.

3.5 Dataset Labeling

The labeled dataset is extracted through the event-based simulation approach presented in Section 3.3.1 on several instances of the dynamic e-ADARP. The information needed to produce the labels is only retrieved during the second phase, i.e. ML-LNS, according to the procedure explained next. Consider an initial sub-problem i, consisting of a routing solution s_{current} and cost $f(s_{\text{current}})$. The initial sub-problem may be modified by applying any of the available destroy-repair operators (d_l, r_n) with $l, n \in \{1, \ldots, m\}$. Since the outcome y_i of each destroy-repair couple depends on the related randomly-drawn destruction level q_i, i.e. the neighborhood size, each destroy-repair operator (d_l, r_n) is applied to s_{current} for M times considering different values of q_i. Namely, the destruction level removes a minimal number of requests (e.g. four requests) and may destroy up to 15%, 25%, 35%, and 45% of the vehicle routes during the M applications of (d_l, r_n). Ideally, M should be large enough in the attempt to de-randomize the statistics associated with the performance of each operator. The objective function improvement between the routing solution before the move (i.e. s_{current}) and the routing solution after the move (i.e. $s^{'}$) from all of the M trials is stored. Objective improvement distributions on the given sub-problem are determined at the end of the M trials for each destroy-repair couple (d_l, r_n). An example of such distributions is given in Figure 3.3. Several statistical measures can be retrieved to characterize the expected objective improvement from the algorithm distributions (e.g. mean, median, mode, percentiles). The choice of the appropriate statistics depend on the goal of the prediction problem. The considered goal is to be able to predict the expected algorithm performances. As such, the labeling considers the mean of the distributions. Note that the mean of any distribution may be negative, as worsening and infeasible solutions are considered during the search. Moreover, note that the mean is highly sensitive to outliers, that is highly positive or negative percentage improvement values. Negative improvements may be either accepted or rejected by the SA approach at any iteration in the search. Specifically, highly infeasible solutions are prevented by the SA approach, namely its temperature and cool rate parameters. However, their evaluation in the mean of the improvement distributions may return expected performances which are heavily skewed towards negative values, i.e. as controlled by the infeasibility penalty cost c_3. In reality, we are interested in retrieving the expected performance a destroy-repair couple may bring, if it successfully moves the current solution to a neighbor solution (i.e. if the move is accepted by SA, that is it works). In other words, we are not interested in specifically estimating the performance of the destroy-repair couple when this last does not move the current solution to a neighbor solution (i.e. it does not work). However, for moves that do work, we may be interested in retrieving the robustness of the destroy-repair couple in moving the current solution towards the estimated neighborhood solution. For example, suppose a destroy-repair couple is expected to bring a high percentage improvement to the current solution. Then, we are also interested in knowing the probability this high percentage improvement realizes again if the given destroy-repair algorithm is re-applied on the current solution. Suppose a similar high-improvement move is very hardly reproducible, then we should penalize the choice of such destroy-repair algorithm with respect to other algorithms which may performe worse but more robustly. As such, we scale the expected performance of each algorithm by the ratio between the total number of accepted neigborhood moves and the total number of attempted moves on the current solution, i.e. M. Furthermore, expected performances which realize less than $\bar{M}$% of the times can be disregarded, as they are very unlikley to re-appear in successive re-applications of the algorithm.

As in the ML-LNS approach, the means from the distributions are successively transformed through an exponential function, and relative performance ratios $p_{d_l,r_n}(i)$ are retrieved. The destroy-repair operator $(d(i), r(i))$ is consecutively drawn according to such ratios. Finally, the next move is computed by re-applying

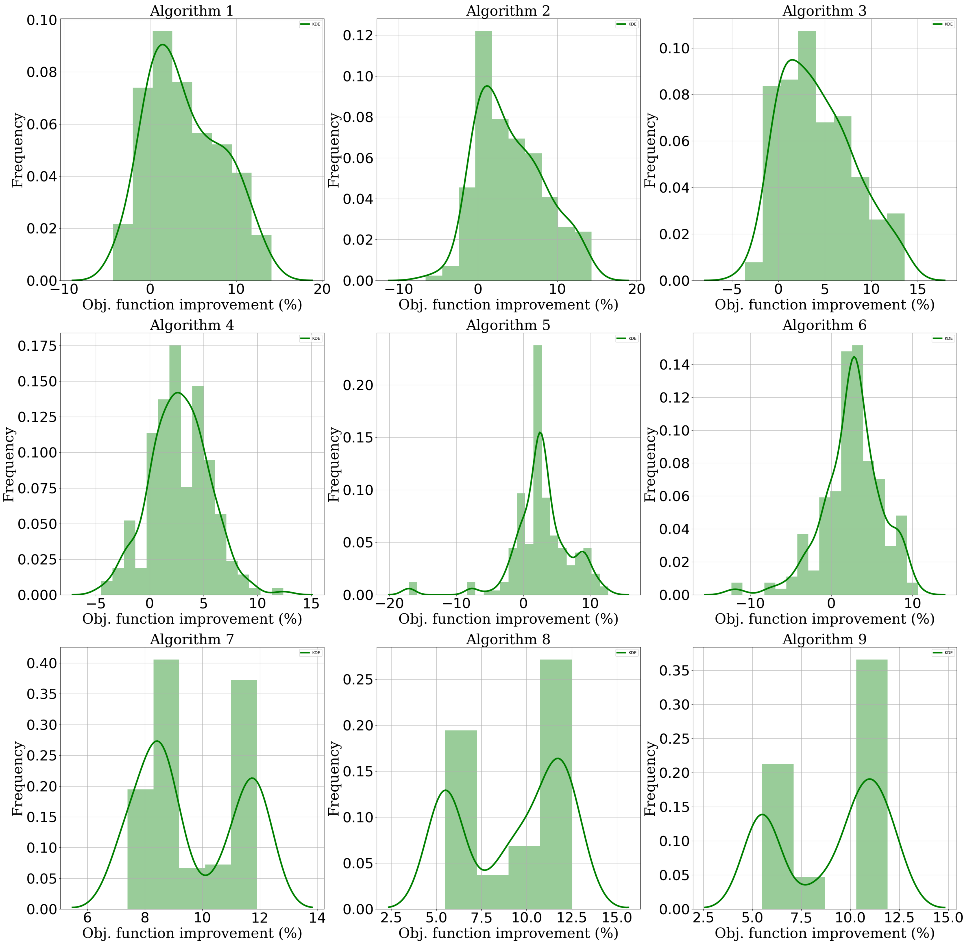


Figure 3.3: Example of the algorithm performance distributions after M trials

the drawn destroy-repair operator $(d(i), r(i))$ on the current solution at iteration i. This last procedure is timed and the search is exited when the search exceeds a time limit of 5 seconds. Note that the proposed labeling approach is computationally heavy, as there are as many neighborhood moves as the number of iterations, destroy-repair operators, and trials per destroy-repair couple.

3.6 Feature Selection

Sub-problems i are characterized by several aggregated features $\mathbf{x}_i$ which are extracted from the instance and routing solution at hand. As noted in Kerschke et al. (2019), the extracted features should be informative (i.e. relevant for distinguishing between problem instances), interpretable (i.e. able to provide insight into instance properties), cheaply computable (i.e. retrievable within milliseconds), generally applicable (i.e. applicable to a broad range of problem instances), and complementary (i.e. un-correlated). Note that the performance of the machine learning approximation specifically adopted in this study, i.e. RF, is not particularly sus-

Table 3.2: Features description

Size and spatial distribution of requests	**Routing solution**
1. Problem size	25. Perc. of used vehicles
2. Perc. of planning horizon elapsed	26.-27. Avg/std number of requests per vehicle
3.-4. Avg/ std radius	28. Avg vehicle travel time
5. Perc. area covered	29. Avg user excess ride time
6.-7. Perc. pickups/dropoffs in convex hull of dropoffs/pickups	30.-31. Avg/std vehicle loads
Upcoming request to be inserted	32.-33. Avg/std vehicle tour lengths
8. Is there a new request to insert?	34.-35. Avg/std number of edges
9.-10. Avg/std distance between vehicle locations and pickup of new request	36.-37. Avg/std charging time per visited charging station
11. Is the dropoff of new request in the convex hull of the planned requests?	38.-39. Avg/std travel time per traversed arc
12.-15. Avg/std distance between new pickup/dropoff and planned requests	40. Perc. of unique travel time arcs
Vehicle current location and states	**Current and past LNS iterations**
16. Perc. of vehicles in convex hull of planned requests	41. Neighborhood size
17.-18. Avg/std distance vehicles from centroid of area	42. Perc. cost difference between current solution and initial solution
19. Avg intra-vehicle distance	43. Perc. cost difference between current solution and last incumbent
20. Perc. of empty vehicles	44. Perc. of requests in request bank
21.-22. Avg/std current vehicle loads	45. Perc. decrease in temperature from initial SA temperature
23.- 24. Avg/std vehicle SOC	

ceptible to correlated input variables. However, correlated variables may affect the interpretability of RF, notably its embedded feature importance evaluation. In the case of vehicle routing problems, most extracted features are necessarily inter-correlated since they are all connected through decision variables and constraints in the problem definition. As such, for vehicle routing problems, feature importance needs to be interpreted in consideration of all possible correlations between the several proposed features. In the dynamic e-ADARP, such aggregated features relate to: (1) the size and spatial distribution of the requests, (2) the upcoming request to be inserted, (3) the vehicle current states and spatial distribution, (4) the routing solution, (4) current and past LNS iterations. Overall, we determine 45 features, summarized in Table 3.2 and explained next.

Size and Spatial Distribution of Requests

We are interested in exploiting features which relate to the demand distribution, and therefore do not depend on the specific routing solution encountered through the ML-LNS. Specifically, we retrieve the total number of requests assigned to the vehicle routes (i.e. 1. the problem size), which were received from the beginning of the planning horizon (2.). Then, we extract the radius (3.-4.), that is the average/standard deviation distance between the planned requests and the centroid defined by the same (Gomes and Selman, 2001). The ratio is a useful metric in detecting whether requests are more or less homogeneously distributed among them. We continue by extracting the ratio between the area of the circle containing all of the assigned requests to the total area covered by the ride-sharing service (5.). This ratio is useful in determining whether the assigned requests are concentrated or scattered within the service zone. Finally, we compute the percentage of pickups/dropoffs contained in the convex hull determined by the dropoff/pickup locations within the vehicle routes (6.-7.). This last feature is helpful in determining whether the instance is characterized by clustered or homogeneously distributed pickup and dropoff locations.

Upcoming Request to be Inserted

A given sub-problem i may be characterized by the existence of an upcoming request which could not be feasibly inserted into the vehicle plans through the first phase of the two-phase metaheuristic approach (Section 3.3.3). The existence of a new un-inserted request highly penalized the objective function, by the cost parameter c_3. Therefore, we start by extracting a binary variable determining whether the sub-problem considers a new request to be inserted (8.). If a new request exists, we characterize the location of the upcoming request by three distinct features. First, we determine the average distance between the vehicle current locations and the pickup of the new request, and its standard deviation (9.-10.). Second, we determine whether the dropoff of the new request lies within the convex hull defined by all requests in the vehicle plans (11.). That is, whether the new request follows the demand pattern defined by the planned requests. Third, we determine the average distance between the pickup and dropoff of the new request to all of the requests in the vehicle plans, and its standard deviation (12.-15.). Specifically, features (11.-15.) help quantify the deviation that may be needed to serve the new request.

Vehicle Current States and Spatial Distribution

We are interested in retrieving the vehicle current states and spatial distribution at the time the sub-problem is re-optimized as this affects the feasibility of neighborhood moves. With respect to the vehicle spatial distribution, we first retrieve the percentage of vehicles contained in the convex hull defined by all of the planned requests (16.). Indeed, a vehicle that just completed its service at a dropoff node may be currently en-route and located far away from the locations of its planned requests. Second, we extract the average distance between the vehicle current locations and the centroid of the area covered by the ride-sharing service, as well as its standard deviation (17.-18.). This feature helps understand the current spatial distribution of the fleet in the service area. Third, we retrieve the average intra-vehicle distance (19.). Differently from the previous feature, intra-vehicle distance helps distinguish cases in which vehicles are homogeneously dispersed in the service area from cases in which group of vehicles are clustered together. With respect to the vehicle current states, we retrieve the percentage of currently empty vehicles (20.), the average current load of the vehicles and its standard deviation (21.-22.), the average vehicle SOC and its standard deviation (23.-24.).

Routing Solution

The iterative solution approach for the e-ADARP explores many solutions which do not differ in terms of the demand distribution and vehicle current states but do differ in terms of route planning. As such, we are particularly interested in extracting valuable information to differentiate between routing solutions. To this aim, we retrieve information on the percentage of utilized vehicles in the solution (25.), the average number of planned requests per vehicle and its standard deviation (26.-27.), the average vehicle travel time (28.) and excess ride time per served request

(29.). Furthermore, we extract aggregate information derived from the sequence in which requests are served within the vehicle plans. Specifically, we retrieve the average vehicle loads along their planned routes and their standard deviation (30.-31.), the average vehicle tour length and its standard deviation (32.-33.), the average number of edges per vehicle tour and its standard deviation (34.-35.), and the average vehicle charging time per visited station and its standard deviation (36.-37.). We further explore features which depend on the vehicle routes and are derived from the travel time cost matrix. Specifically, we extract the average vehicle travel time per traversed arc and its standard deviation (38.-39.), the fraction of unique travel times from the vehicle plans (40.). These last features help determine whether vehicle routes are composed of edges characterized by similar/equal distance (i.e. travel time).

Current and Past LNS Iterations

At any given iteration, it is possible to exploit information from previous iterations of the search. That is, we would like to know the size of the neighborhood that we are currently willing to explore (41.) but also how far we got, in terms of objective function value, from the beginning of the search. To this end, we retrieve the percentage cost difference between the current solution and the initial solution (42.), the percentage cost difference between the current solution and the last incumbent solution (43.), the percentage of requests added to the request bank (44.), and the percentage decrease in temperature for the initial SA temperature (45.). Note that the last two features are related to the degree of infeasibility of the current solution and the current iteration number.

3.7 Numerical Experiments

This work employs real data from 24,000 Uber ride-shares during one-week period in San Francisco, obtained by extracting pickup/dropoff locations and times from published GPS logs.[1]. For the purpose of our tests, we extract 100-request dynamic scenarios. Note that larger problems are not necessarily more difficult to solve, as complexity depends on the size of the encountered sub-problems, which in turn depends on the demand distribution and arrival rate, rather than the size of the dynamic scenarios. We construct a 15-minutes time window around the request pickup/dropoff times and assume instantaneous booking times. The travel time between locations is constructed by converting degree latitutes/longitudes into kilometers, emplying eucledian distance, and assuming constant vehicle speeds of 30 km/h. Maximum ride times is considered fixed for all requests and is set to 30 minutes. We consider a fixed fleet size of 10 vehicles, with a maximal capacity of 15 passengers and a nominal battery capacity of 14.85 kWh. The fleet size has been chosen such that, under the fore-mentioned conditions, 65% of the total demand can be served by a greedy insertion algorithm, on average.

---

[1] Available at `https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata`

The simulation-based optimization procedure described in Section 3.3.2 was implemented in the Julia v. 1.2 (Dunning, Huchette, and Lubin, 2017) and applied to each one of the 244 100-request instances from the Uber dataset. Results were obtained by running each of the simulated scenarios in parallel, by using an Intel Xeon based cluster employing 2 cores per node on Skylake processors running at 2.3 GHz. The second re-optimization phase is triggered any time an incoming request cannot be directly inserted into the vehicle routes and when there are at least four planned requests in the vehicle routes. Solutions that do not include the incoming request or, succesively, any other request from the vehicle routes incur into a penalty cost controlled by parameter c_3, which is set to a very high arbitrary fixed cost. In our work, c_3 is set to two times the average total objective function value from the routes constructed through a simple insertion algorithm.

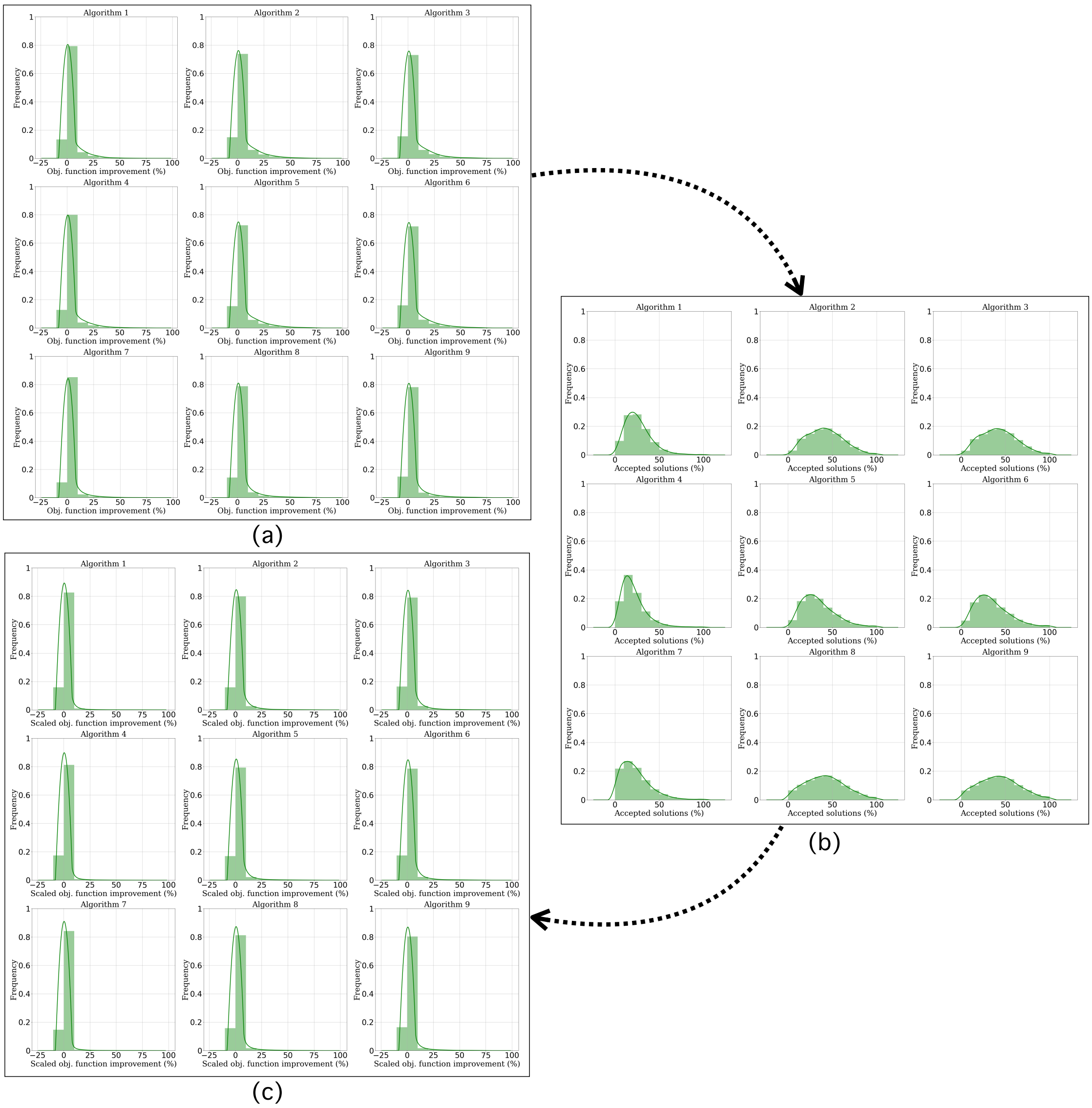
The destroy and repair operators considered in this work closely follow the ones proposed in Ropke and Pisinger (2006). Namely, we adopt all of their destroy (i.e. random, Shaw, and worst) and most of their repair (i.e. basic greedy, regret-2, and regret-3) heuristics. As such, the destroy-repair couples considered in this work, i.e. (d_l, r_n), are obtained by combinations of $d_l \in \{1 : \text{Random (R)}, 2 : \text{Shaw (S)}, 3 : \text{Worst (W)}\}$ and $r_n \in \{1 : \text{Greedy (G)}, 2 : \text{Regret2 (R2)}, 3 : \text{Regret3 (R3)}\}$. Namely, we consider the following LNS algorithms (d_l, r_n): $\{\text{Alg. } 1 : \text{R-G}, \text{Alg. } 2 : \text{R-R2}, \text{Alg. } 3 : \text{R-R3}, \text{Alg. } 4 : \text{S-G}, \text{Alg. } 5 : \text{S-R2}, \text{Alg. } 6 : \text{S-R3}, \text{Alg. } 7 : \text{W-G}, \text{Alg. } 8 : \text{W-R2}, \text{Alg. } 9 : \text{W-R3}\}$. The search is composed of a total of 100 iterations and is notably shorter than what is typically reported in the literature. This choice is motivated by the limited computational time available to make modifications to the vehicle plans in the context of on-line operations (i.e. 5 seconds). Consequently, the search does not have time to explore the solution space for thousands of iterations. For the same reason, the search is limited by a maximal time of 5 seconds. The SA parameters, notably the initial temperature and the cooling rate, are set by assuming that a move that is 5% worse than the current solution is accepted with 50% probability and that such probability converges to zero towards the end of the search. Note that given the high cost associated with infeasible solutions, infeasible solutions are mainly allowed to be explored at the beginning of the search. That is, if the incoming request can only be inserted at the cost of infeasibility for other requests. In this case, the search begins from an infeasible solution and attempts to recover infeasibility in successive iterations. This choice is motivated by the limited time to modify vehicle routes, in the context of our on-line approach. Note that preliminary results had confirmed that over-exploring infeasible solutions in the context of real-time decision making may decrease our ability to find locally optimal solutions.

3.7.1 Labeled dataset

The labeled dataset is constructed according to the principle described in Section 3.5. At each iteration during the search, we employed each of the 9 competing LNS algorithms 100 times. That is, each LNS algorithm tries to explore the neighbor of the current solution a large amount of times. The size of the neighborhood is drawn from a uniform interval which considers removing and reinserting at least four requests and up to 45% of the vehicle routes. For each LNS algorithm, the expected performance is successively computed as the mean of the objective function improvements from the samples which were accepted by SA, out of the 100 trials. In order to characterize the robustness of each LNS algorithm, we scale the expected algorithm performance by the percentage of times neighborhood moves are accepted by SA, i.e. the frequency the algorithm works in moving the current solution in the direction of a new solution. Note that, given that the search is allowed to explore worsening solutions, the expected performance may be negative. As such, we use a smoothened exponential function to transform the algorithm performances, which are consquently normalized returning ratios from which we draw the algorithm to be employed at the current iteration. As explained in Section 3.3.3, the choice of an exponential function is motivated by our need to emphasize differences between improving and worsening algorithms, whose estimates are well-above the average prediction error. The smoothening factor x is selected as a trade-off between two considerations. First, an algorithm showing a relatively high percentage improvement at low rate should not be over-rewarded with respect to other algorithms showing a lower percentage improvement at higher rate. Second, algorithms which perform similarly but still present some differences should be distinguished. As such, after a trial-and-error phase, the smoothening parameter factor x is set to 0.5. Furthermore, distributions with less than 10% accepted trials are not considered and are assigned a highly negative expected improvement.

This procedure resulted in a total of 615,546 labeled datapoints, homogeneously distributed among the 9 competing algorithms. The expected objective function improvement distributions are shown in Figure 3.4(a). As it can be noted, for each algorithm, the expected improvement inhomogeneously distributes between -10% and +90%. Moreover, the algorithms often improve the current solution only marginally, on average. As such, the expected objective improvement for each of the 9 competing algorithms presents many values below 10%. The percentage of times algorithms produce accepted neighborhood moves (i.e. with respect to the SA approach) is shown in Figure 3.4(b). As it can be noted, the percentage of times the current solution is successfully modified into a neighboring solution is highly dependent on the algorithm and vary from a mean of 25% to about 50%. In addition, the robustness of the algorithm seems to be related to the choice of the repair method (e.g. LNS algorithm 1, LNS algorithm 4, and LNS algorithm 7 present similar distributions, which are different from the distributions of LNS algorithm 2, LNS algorithm 5, and LNS algorithm 8). To account for robustness, the expected objective function improvements from Figure 3.4(a) are scaled by the percentage of accepted requests from Figure 3.4(b), producing the scaled objective function improvement distributions in Figure 3.4(c). As it can be noted scaling the objective function improvements by their expectation smoothens the extreme positive values. That is, examples of algorithms producing high improvements at low frequencies.

The produced labeled dataset in Figure 3.4(c) contains examples of moves from the re-optimization policy we aim at learning. Figure 3.5(a) shows the difference



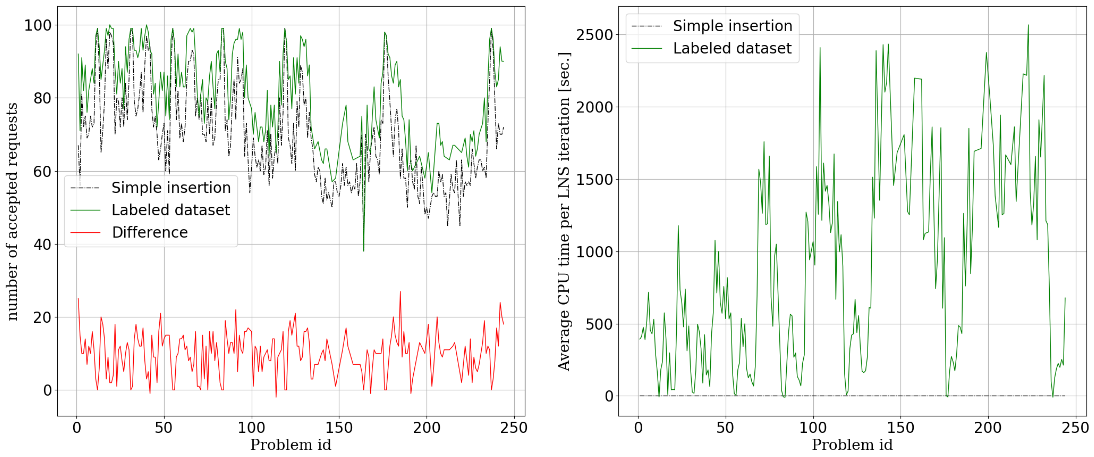
(a)

(b)

(c)

Figure 3.4: Label distributions: (a) Expected objective function improvement, (b) Accepted neighborhood moves by SA, (c) Scaled expected objective function improvement.

in number of accepted requests between the labeled dataset and a simple greedy insertion strategy (i.e. the first phase in the two-phase heuristic approach). As it can be noted, the proposed re-optimization policy has a positive effect on the total number of accepted requests, which increase by up to 27% and by an average of 12% with respect to the insertion policy. In only 8% of the cases, the re-optimization phase may not lead to a higher number of accepted requests and may instead slightly decrease the solution quality (by less than 2%). In fact, optimizing vehicle routes at any given time may not necessarily increase chances to accept more requests later on. As it can be expected, given its computational burden, the extensive methodology used to produce the labeled dataset is not suitable for real-time decision processes, as shown in Figure 3.5(b). Specifically, each iteration of the extensive LNS-based approach used to produce the labeled



(a) (b)

Figure 3.5: Labeled dataset and simple insertion: (a) Difference in terms of number of accepted requests, (b) Average CPU time at each iteration of the search

dataset can take up to about 43 minutes. As such, the re-optimization policy is learned through a statistical learning approach and efficiently re-applied to new on-line problem instances.

3.7.2 Machine Learning Results

A total of 9 random forest regressors are trained on the generated balanced dataset from Figure 3.5(b) by the use of the python ScikitLearn library. As customary in the ML literature, 75% of the labeled set is selected for training, and the remaining 25% for testing. The selected split is the one minimizing the average MAE from 5 random splits on the 244 100-request scenarios. The size of each RF regressor is set to 500 estimators. Note that less estimators may be employed for the treated problem. However, this would impact feature interpretability. On the training dataset, each RF is further optimized with respect to selected hyperparameters, i.e. the maximal number of features and minimal number of samples per split, using grid search and k-fold cross-validation. The optimal parameters are chosen from the results obtained on 5 folds and by considering L-2 loss. Table 3.3 shows performance measures obtained for each of the 9 trained regression models by applying them on the train and test dataset. As shown by the MAE, all models predict the scaled expected objective improvement that should be obtained after the application of any of the 9 algorithms by less than a 2% error. The RMSE shows higher imprecision but still contained within a 5% error. An interesting feature that can be extracted from the MAE and RMSE is that there are classes of algorithms whose average prediction errors seem to be most similar than for others. For example, algorithm 1, 4, and 7 seems to produce similar performance errors, which are different from the ones experienced from the other algorithms. Specifically, algorithm 1, 4, and 7 refer to a greedy insertion heuristic, namely combined with random, Shaw, and worst destruction heuristics. This finding

Table 3.3: Performance measures obtained for the 9 RF regressors on the train and test dataset: Mean absolute error (MAE), root mean square error (RMSE), coefficient of determination (R^2), out of bag error (OOB).

Model	MAE Train-Test	RMSE Train-Test	R^2 Train-Test	OOB
1	0.59 - 1.05	2.11 - 3.30	0.58 - 0.31	0.62
2	1.11 - 1.99	3.15 - 4.71	0.65 - 0.35	0.51
3	1.17 - 2.07	3.31 - 4.85	0.66 - 0.35	0.50
4	0.55 - 0.99	2.12 - 3.29	0.59 - 0.33	0.60
5	1.02 - 1.84	3.05 - 4.52	0.66 - 0.37	0.51
6	1.09 - 1.96	3.24 - 4.71	0.66 - 0.38	0.51
7	0.49 - 0.84	2.15 - 3.33	0.50 - 0.24	0.69
8	0.99 - 1.67	3.23 - 4.53	0.60 - 0.32	0.60
9	1.03 - 1.76	3.36 - 4.70	0.61 - 0.33	0.58

suggests that the prediction problem for regret-2 and regret-3 insertion heuristics may be more challenging than for a greedy heuristic.

The discrepancy between the RMSE and the MSE, as well as the OOB error and the coefficient of determination R^2, suggests that the trained models are over-fitting the training dataset. Over-fit may be due to the high variance that is present in the dataset we employ for training, as shown in Figure 3.5(b). As explained in the previous section, huge differences in percentage improvements are natural in the treated problem, although unrealistically large data differences may be considered as outliers by the trained models. Furthermore, the regressors are trained on the $L-2$ loss (the MSE), which squares the errors, and thus gives more weight to outliers than, for example, $L-1$ loss (the MAE). However, although the MAE is more robust to outliers, it is also computationally much more expensive (i.e. it scales with $Nlog(N)$ as opposed to N for the MSE, where N is the number of datapoints). This complexity is unpractical as it would result in an unrealistic trainig time. Another approach to try and reduce over-fit may consider the use of extra trees regressors, a meta-estimator employing various sub-samples of the dataset and using averaging to control over-fit.

The metrics shown in Table 3.3 are averaged among all of the datapoints in the training/testing dataset. Consequently, we are interested in further analyzing the way the prediction error distributes in the training/testing dataset. Figure 3.6 shows the distribution of the absolute errors between the observed and prediced estimates in the training and testing dataset. As it can be noted, in both cases, we most frequently predict the target values with high confidence. However, in a few cases, we result in high prediction errors up to about +/- 15%, which may explain the reason for the high average prediction errors in Table 3.3.

Finally, note that the ML-LNS uses a smoothened exponential function to transform the predicted expected algorithm performances. As such, even a small mean average error of, for example, 2% may considerably impact our ability to distinguish between algorithms when the predicted estimates from all of the algorithms are around a zero-percent improvement and within the limited range

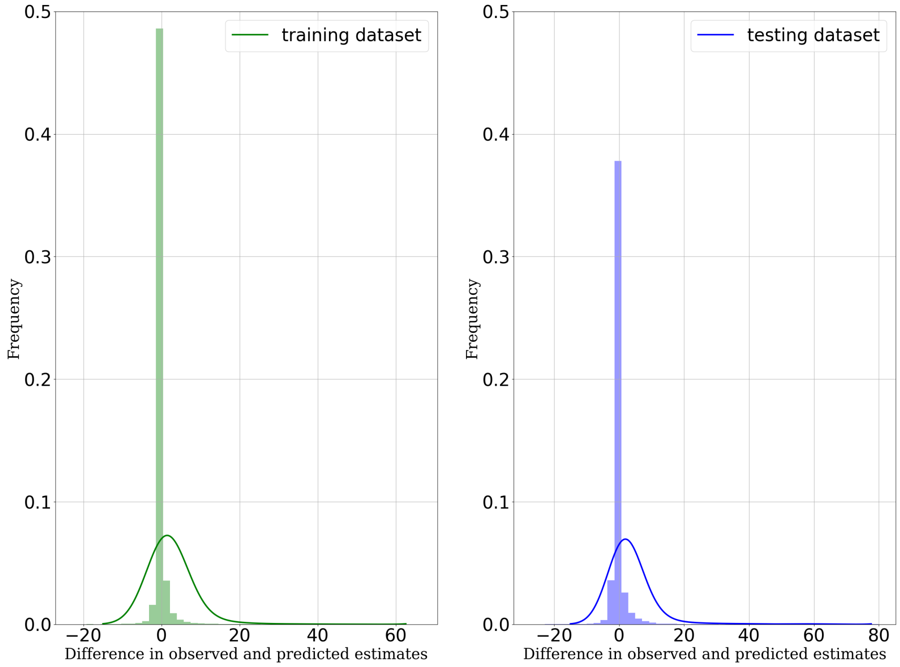


Figure 3.6: Distribution of the absolute errors between observed and predicted estimates in the training and testing dataset

given by the prediction error.

3.7.3 Feature Importance

Figure 3.7 shows the average feature importance among the 9 trained regression models. As a reminder from Section 3.4.2, in RFs feature importance is measured by the average decrease in impurity brought by each feature and between successive nodes in each tree. Impurity may be also interpreted as the average variance reduction brought by each feature. As it can be noticed from Figure 3.7, the two first features decrease the average impurity by at least twice as much the others. These two features relate to the current SA temperature (or current LNS iteration) and the total user excess ride time. Indeed, our ability to improve the current solution depends on history from previous iterations. For example, it is likely that a solution is highly improved at the beginning of the search and only refined at consecutive iterations. Moreover, note that the SA temperature decreases with following iterations. This implies that, at later iterations, accepted solutions can only strictly improve the current objective function. It is therefore reasonable to assume that worsening solutions are most probably accepted at the beginning of the search, as they are controlled by the SA temperature. Solution acceptance depends on the objective function value. This last relates to the second most-important feature, i.e. the user excess ride time. Furthermore, note that the feasibility also affects the objective function and depends on maximum user ride time considerations. The next three most-important features relate to vehicle availability, namely measured by the number of assigned trips per vehicle. Indeed,

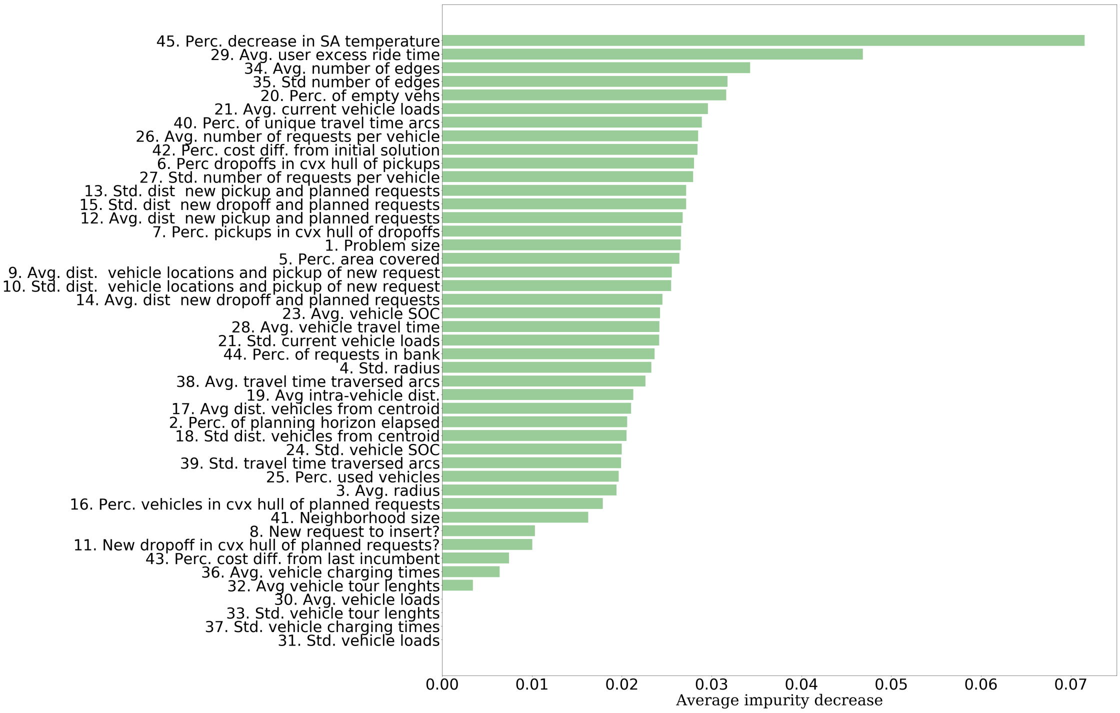


Figure 3.7: Average feature importance

it may be simple to instantaneously deviate vehicles towards a request if the vehicle is currently empty and only have a few planned requests to be served in the future but difficult otherwise. Finally, note that there are features which do not seem to play a significant role in the studied problem. These features relate to the vehicle loads and charging times. Note that, in the studied scenarios, vehicles are over-capacitated and so capacity does not pose an issue for feasibility. Similarly, charging time may not be an issue as vehicles are always allowed to depart from a charging station to serve a request, if this option is feasible. In other words, charging times may only be binding if all vehicles are currently recharging, they have limited battery range and/or there are no other vehicles which could serve the request at a similar operational cost. Finally, it is also interesting to note that the presence of a new request to be inserted does not seem to play a significant role in the ability of the algorithms to improve the current solution. This last observation can be interpreted in two ways. First, all algorithms may find new insertions difficult. Second, all algorithms may find new insertions easy, as new requests can always be inserted at the cost of infeasibility at the beginnig of the search.

3.7.4 ML-LNS Results

The trained models are re-applied to the test dataset, composed of 59 out of 244 100-request instances, and in the context of the ML-LNS explained in Section 3.3.3. The proposed methodology is compared to a random selection algorithm and to ALNS, according to the score updating methodology proposed Ropke and Pisinger (2006). However, given the limited number of iterations to gather statistics, the parameter of the ALNS are set such that operators weights are updated more frequently and react faster to changes in the effectiveness of the operators. Specifically, we set the ALNS reaction factor r to 0.4 and update the operators scores every 5 iterations.

We assume that vehicle route modifications are performed on commercial processors with 8 cores and that computations do not need more than 2 cores. As such, each algorithm (i.e. Random, ALNS, and ML-LNS) is run on 4 parallel optimization processes which terminate after a time limit of 5 seconds. At the end of each parallel re-optimization, the solution resulting in the maximal objective improvement is selected.

In order to compare the outcomes of the Random, ALNS, and ML-LNS re-optimization algorithms, we simulate each one of the tested 59 scenarios 10 times and retrieve partial statistics on their performance. Note that running parallel optimizations on each encountered sub-problem during the search increases the robustness of the statistics, although this last aspect should be confirmed by running more than 10 simulations per scenario.

Figure 3.8 shows distributions on the total number of accepted requests obtained by employing the random, ALNS, and ML-LNS metaheuristics on the tested instances. As it can be noted, independently from the employed algorithm, re-optimizing vehicle routes sensibly increases the total number of accepted requests and by up to about 30%. This also affects the objective function value, whose average improvement per search is shown in Figure 3.9. However, it is hard to clearly distinguish the performance of the metaheuristics, in terms of the average number of accepted requests and average objective function values. In fact, there are instances in which one metaheuristic performs better than the others and vice versa. The distributions on the number of accepted requests and objective function values are furthermore characterized by large standard deviations, although in many cases ML-LNS produces more consistent results (e.g. problems 2, 3, 85, 124, 150, 212). It is interesting to note that in a few cases the metaheuristics return a lower total number of accepted requests with respect to the greedy re-insertion policy (i.e. problems 73 and 78). Indeed, being agnostic of future arrivals, re-optimizing sub-problems at given times may not necessarily maximize the total number of accepted requests at later times. Furthermore, note that differences in sub-problems optimizations imply the simulated scenarios evolve differently. Indeed, as shown in Figure 3.10, each metaheuristic solves a different number and type of sub-problems. Specifically, the sub-problems types differ in terms of demand patterns (as some requests may be served early when considering certain metaheuristics rather than others), the current routing solutions, and vehicle tasks.

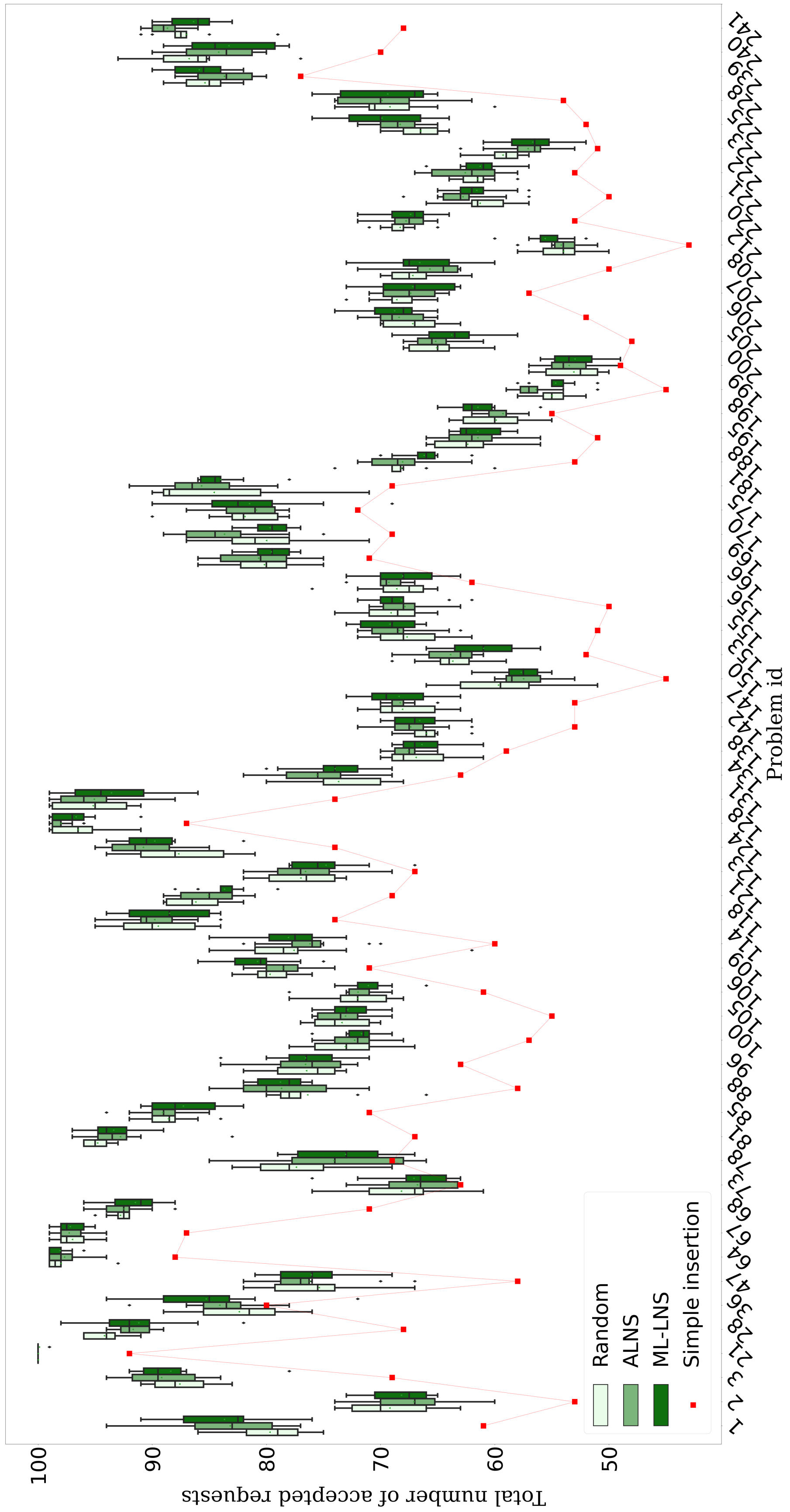


Figure 3.8: Distributions on the total number of accepted requests per algorithm

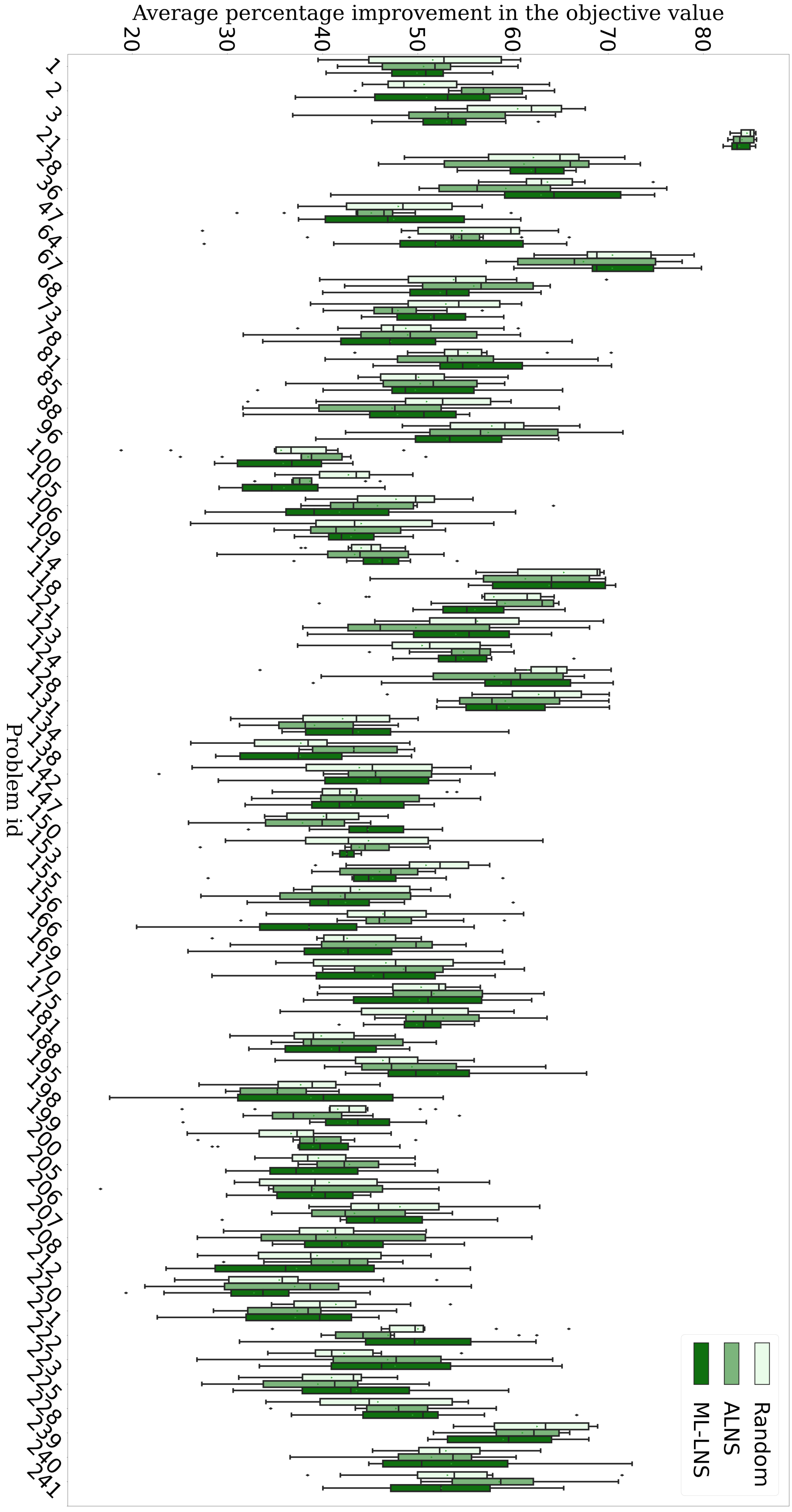


Figure 3.9: Distributions on the average objective function improvement per LNS search

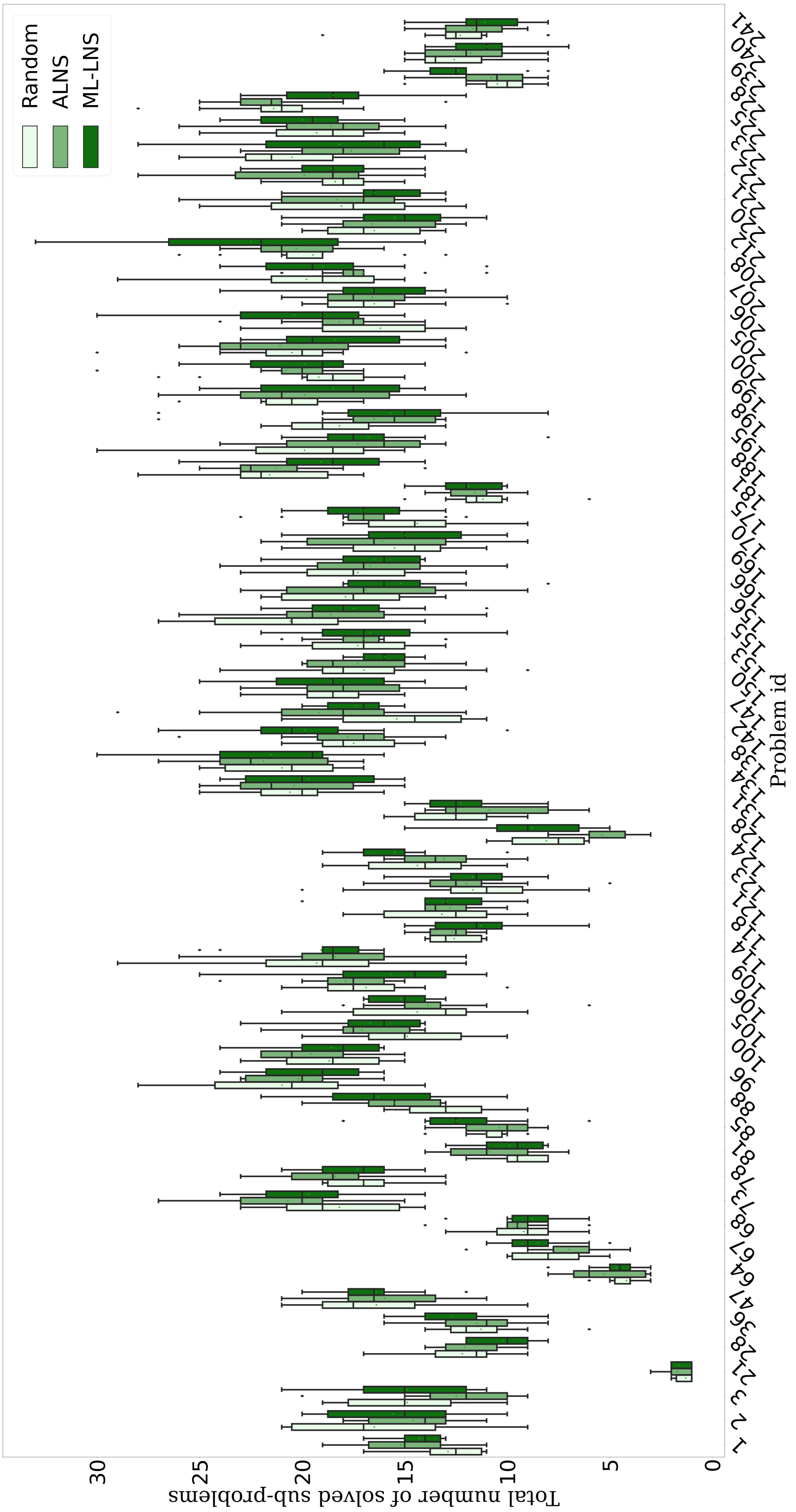


Figure 3.10: Distributions on the number of re-optimized sub-problems per algorithm

Note that each metaheuristic is triggered whenever an incoming request cannot be directly inserted into the vehicle routes. As such, we are interested in analyzing the rate at which algorithms manage to modify the vehicle routes so that the incoming request is feasibly served, as shown in Figure 3.11. It is interesting to note that all algorithms are able to insert upcoming requests with a 60% probability, on average. However, even in this case there is no clear cut between the employed metaheuristics, as they all seem to perform similarly while referring to different problem instances.

Finally, Figure 3.12 shows the average total number of iterations performed by each metaheuristic during the search. These statistics clearly differ between the employed metaheuristics. In particular, the search from the ML-LNS is on average halved with respect to the search for the other metaheuristics. Indeed, at each iteration during the search, the ML-LNS uses of computationally-heavy regression models employing information that needs to be extracted from the routing plans. As such, given the time limit of 5 seconds per search, each iteration of the ML-LNS takes more time to compute with respect to the Random/ALNS metaheuristics. However, as shown before, the limited size of the ML-LNS search does not impact the quality of its results. In fact, they are comparable to the results obtained by the Random/ALNS metaheuristics, although the latter are given twice as many opportunities to explore neighboring solutions. As such, it can be inferred that the ML-LNS makes informative decisions when moving between the few explored neighborhoods. Note that the 9 employed regression models in the ML-LNS are the RF regressors composed of a total of 500 estimators, i.e. trees, as explained in Section 3.7.4. In order to decrease the total computational time per ML-LNS iteration, it could be sufficient to decrease the total number of estimators employed during the search (e.g. from 500 to 100 estimators). However note that this may come at the cost of estimation precision and, consequently, valuable information to the ML-LNS. As such, current work is investigating the right trade-off between computational complexity, measured by the number of employed estimators, and solution quality.

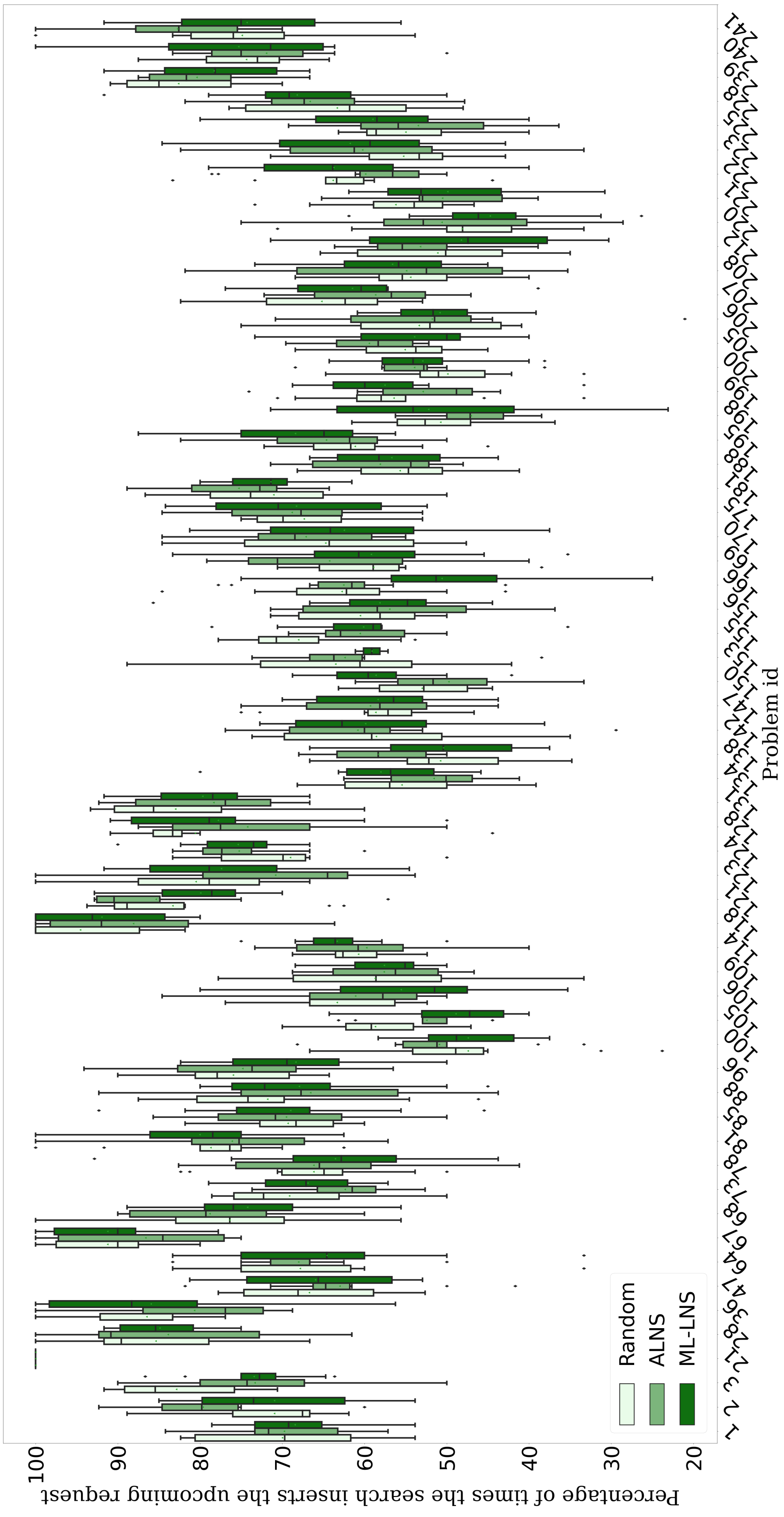


Figure 3.11: Distributions on the percentage of times algorithms introduce the upcoming request during the search

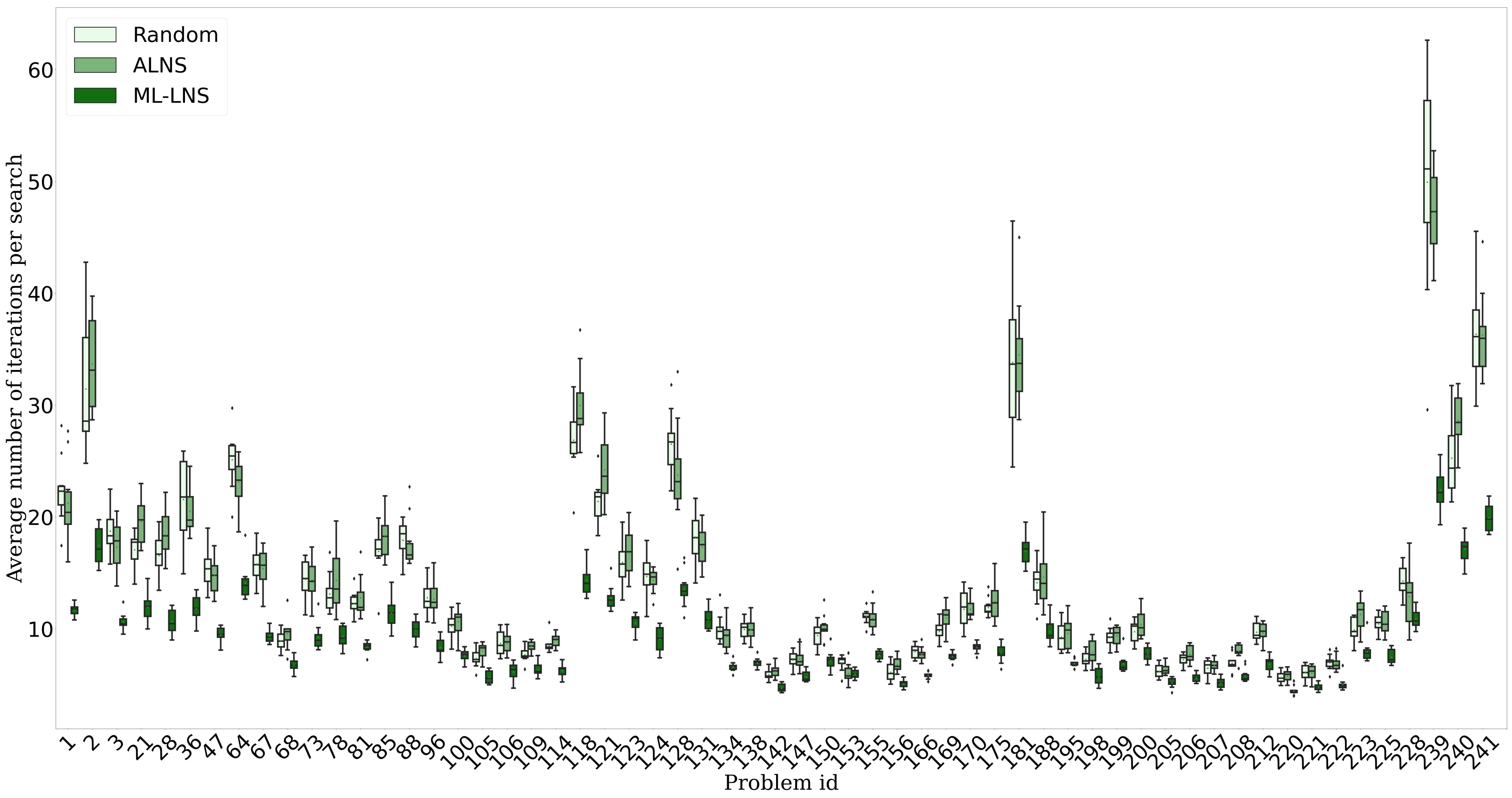


Figure 3.12: Distributions on average number of LNS iterations per algorithm

3.8 Summary

This work proposes a new extension to the family of large neighborhood search metaheuristics for the dynamic electric autonomous dial-a-ride problem (ML-LNS). The proposed metaheuristic employs a machine learning approach to predict the performance on the objective function of 9 destroy-repair couples for each iteration during the search. The training dataset is composed of a large number of examples of LNS moves and their respective expected objective improvement. We have trained 9 different models through a random forest regression approach. Results show that the learned models are capable of reproducing the observed data with high confidence. The models have been re-employed on the test dataset in the context of a simulation-based optimization approach. Results show that ML-LNS is a competitive alternative to state of the art metaheuristics. Current work is extending the results by analyzing whether the ML-LNS may outperform the state of the art under specific circumstances.

There are three main drawbacks in the adoption of a machine learning-based mechanism rather than an adaptive mechanism. First, historical data from LNS iterations on past problems may not be readily available and needs to be specifically produced for the given routing problem, through computationally-expensive simulations. Second, instances from the given routing problem need to be characterized by features that can only be selected through expert knowledge. Third, the adopted ML predictor needs to be appropriately calibrated through computationally expensive experiments during the off-line training phase. Future work may include: (1) The adoption of a look-ahead policy to partially guide the optimization algorithm by future demand arrivals, (2) An analysis on the impact of the re-optimization frequency, (3) A comparison between service levels when adopting large and local neighborhood moves, (4) The adoption of other classification frameworks from machine learning literature.

4

Scheduling Algorithm and Battery Management Heuristic for the Electric Autonomous Dial-a-Ride Problem

This chapter is based on the article:

- C. Bongiovanni, M. Kaspi, and N. Geroliminis (2020). “Scheduling Algorithm and Battery Management Heuristic for the electric Autonomous Dial-a-Ride Problem”. Working paper

4.1 Introduction

In a vehicle routing context, scheduling algorithms prove time feasiblity of vehicle routes. Specifically, they consider feasible routes in terms of capacity and precedence constraints and are designed to provide feasible decisions about time-related variables (i.e. service start times and waiting times at each node). In the standard DARP, such decisions need to respect time-window and maximum-ride-time constraints. In the e-ADARP, scheduling decisions need to further respect battery-management and charging constraints. As such, scheduling algorithms for the e-ADARP need to find the right trade-off between battery feasibility and the level of service, measured by the user excess ride time. Note that at the scheduling point, feasible routing decisions are already taken, and therefore the objective components that solely depend on the routing decisions do not need to be considered.

In the DARP literature, various heuristics have been proposed to deal with the scheduling of routes. Cordeau and Laporte (2003) propose an 8-step procedure setting the earliest start time at each vertex in the route and using forward slack times to delay the start times at pickup locations in vision of maximum ride time constraints (Savelsbergh, 1992). Parragh et al. (2009) observe that adopting a sequential approach to avoid maximum ride time violations does not necessarily minimize the total user excess ride time in the schedules. In fact, delaying the service start time at a given pickup location may decrease the excess ride time of the specific request but increase the excess ride time for other requests in the route. As such, Parragh et al. (2009) modify the procedure in Cordeau and Laporte (2003) by adapting the computation of forward slack times such that increases in the user excess ride time of any request in the route is avoided. As a result, the returned feasible schedules minimize the total user excess ride time at the expense of incorrect infeasibility declarations. This last aspect is a drawback that is tackled in Molenbruch et al. (2017), who propose a procedure starting by considering a possibly travel-time infeasible schedule setting the excess ride time of each user at its lowest bound. Infeasibility relates to travel time shortages between successive nodes and is succesively recovered by shifting service start times such that the total user excess ride time is minimized.

The proposed scheduling algorithms from literature are not proven to provide excess-time optimal solutions. Other than for its theoretical reasons, providing an excess ride-time-optimal scheduling procedure allows to avoid to explicitly express all time-related decision variables and constraints in dial-a-ride problems in which the objective function includes level of service aspects (e.g. in the DARP extensions proposed by Parragh (2011) and Molenbruch et al. (2017)). Instead, all time-related constraints, affecting the objective function, could be separated as needed, i.e. through lazy constraints. However, note that removing the time-related decision variable and constraints from the formulation may result in weakened initial lower bounds. As such, time-related lazy constraints need to be separated through efficient valid inequalities. Furthermore, a polynomial-time optimal procedure would also be beneficial in any meta-heuristic applied for the e-ADARP, as it will allow properly evaluating a given sequence of nodes.

In this Chapter, we propose an algorithm which is proven to provide excess-time optimal solutions and we provide a heuristic procedure to assign charging times at visited stations. Indeed, a schedule minimizing excess ride time may not be battery feasible, and vice versa. The procedure builds on two main observations: (1) minimizing excess-time while respecting time-window and user-ride-time constraints is in fact equivalent to assigning the right amount of waiting time at all nodes in the routes, (2) ensuring battery feasibility is in fact possible by recharging as much as possible at the visited facilities, as early as possible. The algorithm is tested against optimal scheduling solutions from a linear program on the e-ADARP instances from Bongiovanni, Kaspi, and Geroliminis (2019) and presented in Chapter 2.

The rest of the chapter is organized as follows: Section 4.2 introduces the excess-ride-time scheduling problem, Section 4.3 provides the route evaluation procedure and its optimality proof, Section 4.4 explains the heuristic procedure to provide battery-feasibility. Finally, Section 4.5 provides numerical experiments comparing the exact solutions obtained through the employment of a linear program and the scheduling procedure and Section 4.6 summarizes the main concepts of this

Chapter and provides an overlook to future research.

4.2 Scheduling Problem

Consider a predetermined route sequence $\mathcal{I}$ of M nodes, which includes pickup locations, dropoff locations, and potentially some charging stations. Without loss of generality, assume that the sequence satisfies routing constraints, as well as precedence and load constraints. Note that the given sequence may not necessarily satisfy all timing and battery management constraints. Then, the optimization problem consists in scheduling the service start times in the sequence as to minimize the total user excess ride time, guarantee battery and maximum ride time feasibility. Note that user excess ride time is defined as the delay users experience for sharing rides, in comparison to a taxi service.

Start by considering a specific sub-sequence $\bar{\mathcal{I}}$ of $\bar{M} \leq M$ nodes, which is one of the derived sub-sequences obtained by splitting $\mathcal{I}$ by the visited charging stations. Without loss of generality, assume that the visited charging station at the beginning of $\bar{\mathcal{I}}$ represents an origin depot and the charging station at the end of $\bar{\mathcal{I}}$ represents a destination depot. Furthermore, denote by $i \in \{1, \ldots, n\}$ the set of requests contained in sequence $\bar{\mathcal{I}}$, P_i the set of pickups, and D_i the set of dropoffs. Pickup locations are characterized by loads l_{P_i} and dropoff locations by loads $-l_{P_i}$. Furthermore, all locations feature service times d_i, direct travel times $t_{i,j}$, and time windows $[arr_i,\ dep_i]$ limiting the time at which service may start. Note that, in vision of user maximum ride times u_{P_i}, it is possible to set time windows around the pickup locations and derive the time windows around the corresponding dropoff locations, and vice versa. The goal is to be able to optimize $\bar{\mathcal{I}}$ with respect to the total excess ride time by setting optimal service start times T_i with $i \in \{1, \ldots, \bar{M}\}$ in vision of maximum ride time and time window constraints. As such, the scheduling problem for sub-sequence $\bar{\mathcal{I}}$ can be stated as the following linear program (LP1):

$$(LP1)\ \ \min \sum_{i\in\{1,\ldots,n\}} (T_{D_i} - T_{P_i} - d_{P_i} - t_{P_i,D_i}) \tag{4.1}$$

Subject to:

$$T_i + t_{i,i+1} + d_i \leq T_{i+1} \quad \forall i \in \{1, 2, \ldots, \bar{M}-1\} \tag{4.2}$$

$$T_{D_i} - T_{P_i} - d_{P_i} \leq u_{P_i} \quad \forall i \in \{1, \ldots, n\} \tag{4.3}$$

$$arr_i \leq T_i \leq dep_i \quad \forall i \in \{1, 2, \ldots, \bar{M}\} \tag{4.4}$$

Where, the objective function (4.1) minimizes the excess ride time for each request in $\bar{\mathcal{I}}$, constraints (4.2) set the service start times between consecutive nodes, while constraints (4.3)-(4.4) impose maximum ride time and time window constraints respectively.

Note that time windows $[arr_i,\ dep_i]$ can be tightened in light of the travel times and service times between consecutive nodes. As such, by sequentially inspecting the sequence, it is possible to calculate the earliest time ET_i and latest time LT_i at which service can start at node i by using the following recursive formulas:

$$ET_i = \max\{arr_i, ET_{i-1} + t_{i-1,i} + d_i\} \quad \forall i \in \{2, \ldots, \bar{M}\}, ET_1 = arr_1 \tag{4.5}$$

$$LT_i = \max\{dep_i, LT_{i+1} - t_{i,i+1} - d_i\} \quad \forall i \in \{1, \ldots, \bar{M} - 1\}, LT_{\bar{M}=dep_{\bar{M}}} \tag{4.6}$$

Hence, a tighter formulation to LP1 can be obtained by substituting constraints (4.4) with:

$$ET_i \leq T_i \leq LT_i \quad \forall i \in \{1, 2, \ldots, \bar{M}\} \tag{4.7}$$

Service start time T_i at node i depends on the initial departure time from the depot, the total travel time up to i, the total service time spent serving nodes before i, and the total vehicle waiting time up to i. That is:

$$T_i = T_1 + \sum_{j=1}^{i-1} t_{j,j+1} + \sum_{j=1}^{i-1} d_j + \sum_{j=1}^{i} w_j \tag{4.8}$$

Here, w_i denotes the waiting time at node i. Note that, service may start as soon as possible without penalizing the objective function, i.e. $T_1 = ET_1$. With such representation of service start times, the objective function (4.1) can be re-written as follows:

$$\min \sum_{i\in\{1,\ldots,n\}} (T_1 + \sum_{j=1}^{D_i-1} t_{j,j+1} + \sum_{j=1}^{D_i-1} d_j + \sum_{j=1}^{D_i} w_j) - (T_1 + \sum_{j=1}^{P_i-1} t_{j,j+1} + \sum_{j=1}^{P_i-1} d_j + \sum_{j=1}^{P_i} w_j) - d_{P_i} - t_{P_i,D_i} =$$

$$\min \sum_{i\in\{1,\ldots,n\}} (\sum_{j=P_i}^{D_i-1} t_{j,j+1} + \sum_{j=P_i}^{D_i-1} d_j + \sum_{j=P_i}^{D_i} w_j - d_{P_i} - t_{P_i,D_i}) \tag{4.9}$$

Since travel times between consecutive nodes and the service times are deterministic parameters, minimizing the objective function (4.9) is equivalent to:

$$\min \sum_{i\in\{1,\ldots,n\}} \sum_{j=P_i+1}^{D_i} w_j = \min \sum_{i=1}^{\bar{M}} L_i w_i \tag{4.10}$$

Where $L_i = \sum_{j=1}^{i-1} l_j$ represents the vehicle load up to node i.

Equivalent to the re-writing of the objective function, constraints (4.7) can be re-defined through (4.8) as follows:

$$\sum_{j=1}^{i} w_j \geq ET_i - \sum_{j=1}^{i-1} t_{j,j+1} + \sum_{j=1}^{i-1} d_j - ET_1 \quad \forall i \in \{1, 2, \ldots, \bar{M}\} \tag{4.11}$$

$$\sum_{j=1}^{i} w_j \leq LT_i - \sum_{j=1}^{i-1} t_{j,j+1} + \sum_{j=1}^{i-1} d_j - ET_1 \quad \forall i \in \{1, 2, \ldots, \bar{M}\} \tag{4.12}$$

Note that these constraints provide lower and upper bounds to the total waiting time amount that needs to be distributed between i and all nodes preceding i. As such, the linear program (LP1) can be equivalently re-defined as the following linear program (LP2):

$$(LP2)\ \ \min \sum_{i=1}^{\bar{M}} L_i w_i \tag{4.13}$$

Subject to:

$$\sum_{j=1}^{i} w_j \geq ET_i - \sum_{j=1}^{i-1} t_{j,j+1} - \sum_{j=1}^{i-1} d_j - ET_1 \quad \forall i \in \{1, 2, \ldots, \bar{M}\} \tag{4.14}$$

$$\sum_{j=1}^{i} w_j \leq LT_i - \sum_{j=1}^{i-1} t_{j,j+1} - \sum_{j=1}^{i-1} d_j - ET_1 \quad \forall i \in \{1, 2, \ldots, \bar{M}\} \tag{4.15}$$

$$\sum_{j=i+1}^{D_i} w_j \leq u_i - \sum_{j=i}^{D_i-1} t_{j,j+1} - \sum_{j=i+1}^{D_i-1} d_j \quad \forall i \in \mathcal{P} \tag{4.16}$$

Remark that constraints (4.2) from (LP1) are guaranteed by the definition of equation (4.8), which now composes constraints (4.14) and (4.15). The right-hand side of (4.14) and (4.15) represent the minimal total waiting time that must be assigned up to node i, and the maximal total waiting time that can be assigned up to node i, without violating the time windows at sucessive nodes in the sequence. For convenience, denote the right-hand side of constraints (4.14) and (4.15) by Δ_i and Θ_i respectively. Using equations (4.5) and (4.6) we have that $\Delta_i \leq \Delta_{i+1}\ \forall i \in \{1, 2, \ldots, M-1\}$ and $\Theta_i \leq \Theta_{i+1}\ \forall i \in \{1, 2, \ldots, M-1\}$. That is, the total minimal and maximal waiting time that needs to be distributed in the sequence may only increase between consecutive nodes. As such, note that the total minimal waiting time Δ_M, i.e. at the destination depot, represents a waiting time amount that cannot be avoided in any feasible solution. Finally, the objective of the problem reduces to optimally distribute Δ_M among all nodes $\{1, \ldots, M\}$ in consideration of the total load L_i at each node in the sequence, which impacts the total user excess ride time.

4.3 Scheduling Procedure

In order to solve (LP2), we propose the procedure reported in the pseudo-code from Algorithm (2) and explained next. The algorithm proceeds in the sense of the sequence and checks that, for each encountered node $i \in \{1, \ldots, \bar{M}\}$, a minimal total waiting time Δ_i has been assigned up to node i. If the algorithm detects a total waiting time shortage at node i, i.e. $\sum_{k=1}^{i} w_k < \Delta_i$, the total waiting time at i and its preceding nodes needs to be increased. Given that the objective function depends on the total vehicle load L_i, the algorithm starts by considering adding waiting time to nodes $j \leq i$ featuring the lowest minimal total load up to i, i.e. $j = \operatorname{argmin}_{k \in \{1,\ldots,i\}} L_k$. Note that if multiple nodes featuring an equivalent minimal total load exist, the first node can be selected without loss of generality. For node j, the total waiting time can be feasibly increased by a maximum amount δw_j defined by constraints (4.15) and (4.16). That is, while deciding upon an increment in waiting time at node j, one needs to check that excess-ride time constraints are not violated for requests whose pickups precede j and whose dropoffs follow j. Furthermore, in order to guarantee time-window feasibility of the whole sequence, one needs to check that an increment in waiting time at j does not exceed the maximal waiting time that can be assigned to up to node j, i.e. $\Theta_j - \sum\limits_{k=1}^{j} w_k$. Finally, waiting time at node j can be incremented by δw_j, which is computed as the minimum between the amount defined by excess-ride time constraints, time-window constraints, and the total waiting time shortage defined by $\Delta_i - \sum\limits_{k=1}^{i} w_k$. After the update of w_j, if node j has reached its maximum waiting time limit by updating Δ_i, that is $\delta w_j < \Delta_i - \sum\limits_{k=1}^{i} w_k$, node j is removed from the list Ω of potential nodes whose waiting time may be further increased. If $\sum\limits_{k=1}^{i} w_k < \Delta_i$, i.e. there is still a total waiting time shortage at i, the total waiting time is increased at the next node $\bar{j}$ up to i featuring the second lowest total vehicle load. This iterative process terminates as soon as $\sum\limits_{k=1}^{i} w_k \geq \Delta_i$, that is when sufficient waiting time has been assigned to i and all of its preceding nodes. In this case, the algorithm moves inspecting $i + 1$ and up to the end of the sequence. Note that if the waiting time at all of the nodes preceding i have been updated, i.e. $\Omega = \emptyset$, but there is still a waiting time shortage at i, i.e. $\sum\limits_{k=1}^{i} w_k < \Delta_i$, the algorithm prematurely terminates and the sequence is proven infeasible. Instead, if at the end of the whole process, $\Delta_{\bar{M}}$ has been feasibly assigned, the algorithm terminates with a basic feasible solution. Note that the procedure results in a worst-time complexity of $O(\bar{M}^2)$, since in the worst case, the step reported in line 4. of Algorithm (2) may be executed $\bar{M}$ times and the procedure in line 6. contains at most $\bar{M}$ components.

In order to demonstrate that the obtained solution is optimal, it is sufficient to show that there does not exist a neighboring basic feasible solution that strictly improves the objective function (4.13). By construction, since $\sum_{i=1}^{\bar{M}} w_i = \Delta_{\bar{M}}$, any decrease of w_i at some node i would require an increase of waiting time at some other node j. Node j may either precede i (i.e. $j < i$) or succeed i (i.e. $j > i$). Next, both cases are analyzed:

- $j < i$: If decreasing the waiting time at i and increasing it at j by the same amount results in a decrease of the objective function value, this means that $L_j < L_i$. However, in this case, the algorithm would have exploited all the waiting at j before considering i. Therefore, increasing the waiting at a preceding node is either infeasible or would not result in a decrease in the objective function.

- $j > i$: Decreasing the waiting time at i and increasing it at j is feasible if and only if we have $\sum_{k=1}^{l} w_k > \Delta_l \ \forall i \leq l < j$. By construction, this may be the case if $L_i \leq L_j$. This implies that such a transition does not result in an improvement of the objective function value.

Algorithm 2: Excess-time optimal scheduling algorithm

```
Input: vehicle route sequence ($\bar{I} = \{1, \dots, \bar{M}\}$), pickups $P_i$, dropoffs $D_i$, earliest start times $ET_i$,
       latest start times $LT_i$, maximum ride times $u_{P_i}$, service durations $d_i$, travel times $t_{i,j}$
Output: Waiting times $w_i$ with $i \in \bar{I}$, feasibility check
initialize $w_i. = 0 \;\; \forall i \in \{1, \dots, \bar{M}\}$;
initialize $\Omega = \emptyset$ ;
while check = true do
    for $i \to 1$: do
        Update: $\Omega = \Omega \cup \{i\}$ ;
        while $\sum_{k=1}^{i} w_k < \Delta_i$ do
            Set: $j = \mathrm{argmin}_{k \in \Omega} L_k$ ;
            Compute: $\delta w_j = \min\{\min_{k \in \{P | k \leq j \;\&\; n+k \geq j\}} u_k - \sum_{l=k}^{(n+k)-1} t_{l,l+1} - \sum_{l=k+1}^{(n+k)-1} d_l - \sum_{l=k+1}^{n+k} w_l;\; \Theta_j - \sum_{k=1}^{j} w_k;\; \Delta_i - \sum_{k=1}^{i} w_k\}$;
            Set: $w_j = w_j + \delta w_j$;
            if $\delta w_j < \Delta_i - \sum_{k=1}^{i} w_k$ then
                Update: $\Omega = \Omega \setminus \{j\}$;
            if $\sum_{k=1}^{i} w_k \geq \Delta_i$ then
                break;
            if $\sum_{k=1}^{i} w_k < \Delta_i \;\&\; \Omega = \emptyset$ then
                check = false
                break;
```

To conclude, for the obtained basic feasible solution, there is no neighboring feasible solution which strictly improves the objective function. Therefore, the obtained solution is optimal. The implications can be finally summarized as follows:

1. There exists an optimal solution minimizing completions time, i.e. in which service at end depot starts at the earliest time possible $T_{\bar{M}} = ET_{\bar{M}}$.
2. There exists an optimal solution in which service at the first node begins as late as possible, i.e. $T_1 = LT_1$. In fact, given that at the beginning of the sequence the vehicle is empty, i.e. $L_1 = 0$, waiting in the first node can be maximized without affecting the objective function, i.e. $w_1 = \Theta_1$.
3. Building on the previous point, there exists an optimal solution in which service at the first node begins at any time between ET_1 and LT_1 without increasing the excess ride time of any request in the route.
4. For the case in which $\Theta_1 \geq \Delta_{\bar{M}}$, the entire necessary waiting can be done at the first node. As a result, we obtain multiple solutions in the following structure: $\Theta_1 - \Delta_{\bar{M}} \leq w_1 \leq \Theta_1$ and $w_i = 0 \;\; \forall i \in \{2, \dots, \bar{M}\}$.
5. For the case $\Theta_1 \leq \Delta_{\bar{M}}$ there exists an optimal solution where service at the first node begins as late as possible at the first node ($w_1 = \Theta_1$) and begins as early as possible at the last node ($\sum_{i=1}^{\bar{M}} w_i = \Delta_{\bar{M}}$).

Algorithm 3: Battery management algorithm

```
Input: vehicle route sequence ($I = \{1, \dots, \bar{M}\}$), charging facilities $\{s_1, \dots, s_N\}$, charging rates $\alpha_i$,
       discharging rate $\beta$, travel times $t_{i,j}$, earliest start times $ET_i$, latest start times $LT_i$, SOC $B_i$
Output: Charging times $E_i$ with $i \in \{s_1, \dots, s_N\}$, Boolean battery feasibility check
initialize $check = true$;
initialize $B_1 = B_0$ ;
while $check = true$ do
    for $i \to 2$: do
        Compute: $B_i = B_{i-1} - \beta \times t_{i,j}$;
        if $B_i < 0$ then
            the sub-sequence is infeasible $\to$ $check = false$;
        if $i \in \mathcal{S}$ then
            Set: $E_i = \min\{LT_{i+1} - ET_i; (Q - B_i)/\alpha_i\}$;
            Set: $B_i = B_i + E_i$;
```

4.4 Battery Management Heuristic

Next, after computing excess-time optimal schedules for all of the sub-sequences in $\mathcal{I}$, one needs to show that the schedule is battery-feasible. Indeed, the excess-ride-time optimal scheduling procedure disregards battery considerations which are instead part of the e-ADARP. Battery-feasibility aspects can be implemented by integrating battery-related decision variables and constraints into (LP1), as presented in section 2.2.2 of Chapter 2.

Note that vehicle battery levels can be seen as an inventory which can only decrease with traveling. Then, for feasibility purposes, it is always better to recharge as much as possible, as early as possible. If the schedule that maximizes battery recharging does not satisfy the imposed battery management constraints, the given sequence and excess-time optimal schedule is declared infeasible. Following implications 4. and 5. from the previous page, if battery-feasible schedules do exist, at least one of them exhibits the total user excess-ride time computed through the proposed scheduling algorithm. Note that the proposed recharging procedure is a heuristic since it cannot be formally proven optimal (e.g. maximizing the charging time at station i may decrease opportunities to recharge more at following charging stations). Denote by N the number of charging stations contained in sequence $\mathcal{I}$. Let Q represent the nominal capacity of the e-AVs β the discarging rate, and B_i the battery inventory level at locations $i \in \mathcal{I}$, and α_s the charging rate at charging facilities $s \in \mathcal{S}$.

The procedure in Algorithm 3 is proposed for battery management. The algorithm sequentially computes the battery inventory between successive nodes in consideration of the initial battery level B_1, the battery discharge rate β, and the total travel time up to i. Note that battery discharge can be equally computed by energy consumption models (Goeke and Schneider, 2015; Pelletier et al., 2017a). During this iterative process, when a charging facility is encountered, its maximal recharging time is bound by:

1. The difference between the service start time at the current node and the latest service start time at the following node,

2. The time needed to fully recharge.

To ensure feasibility, the recharging time at the station needs to be set to the minimum of the two. Furthermore, earliest start times E_i and latest start times L_i are computed by taking into account the eventual waiting times obtained from the scheduling algorithm. The procedure prematurely terminates only if a node i features a negative battery inventory, after which the route is declared battery-infeasible.

4.5 Numerical Experiments

Numerical experiments are performed on the second set of instances presented in Chapter 2, i.e. the instances extracted by real data from Uber Technologies Inc. Specifically, we extract the routing solutions obtained for those instances and solve the scheduling problem through a linear program and the route evaluation procedure proposed in this Chapter. As explained in Section 4.4, the linear program is obtained by supplementing (LP1) with charging and battery management constraints/decision variables as proposed in Chapter 2. The numerical experiments are implemented in Julia v.1.2 on a 2.50 GHz Intel(R) Core(TM) i7 processor with 16 Gb of RAM. The LP is implemented in the JuMP modeling language (Dunning, Huchette, and Lubin, 2017) and solved with Gurobi v. 9.0.

Table 4.1 shows a comparison between the scheduling solution obtained through the use of the LP and the route evaluation procedure. The first column indicates the instance name, whereas the second column indicates the total number of visited charging stations along the vehicle routes. As a reminder from Chapter 2, the following convention is used for the instance names: <u><number of vehicles>-<number of customers>-<minimum battery inventory ratio at the destination depot>,“u" is used to refer to the instances adapted from real data from Uber Technologies Inc. Furthermore, the reader is reminded that the minimum battery inventory ratio is used to compute the minimal battery SOC all vehicles need to have at the destination depot, i.e. a minimal battery ratio of 0.7 means vehicles need to have at least 70% of their nominal battery capacity at the destination depot. The following columns in Table 4.1 show the total vehicle charging time (in minutes), the total vehicle waiting time (in minutes), the objective function value (i.e. total user excess ride time, ERT, in minutes) and the total computing time (CPU, in seconds) for the scheduling solutions obtained by the use of the LP and the route evaluation procedure.

As it can be noted by the results in Table 4.1, the route evaluation procedure provides equivalent objective function values to the LP, showing that the proposed procedure returns optimal solutions for the given problems. Although the difference in CPU time between the two approaches is almost negligible for the tested problems, the route evaluation procedure is on average faster and reduces the CPU time by about a factor of 10, on average. Savings in terms of computing time might be even more significant when considering larger problem instances or when several static problems need to be solved in real time, e.g. as in the dynamic e-ADARP presented in Chapter 3.

With respect to the total vehicle waiting time, it can be noted that the route

Table 4.1: Comparison between the scheduling solution of the LP and the route evaluation procedure. In order of appearance: instance name, total number of visited stations, total vehicle recharging time from the LP solution [min], total vehicle waiting time from the LP solution [min], objective function value (total user excess ride time) from the LP solution [min], total cpu time to retrieve the LP solution [sec]; analogous results for the route evaluation procedure.

	Linear Program					Route evaluation procedure			
Name	**Stations**	**Charging**	**Waiting**	**ERT**	**CPU**	**Charging**	**Waiting**	**ERT**	**CPU**
u2-16-0.1	1	5.72	155.47	0.00	0.022	11.12	132.70	0.00	0.001
u2-16-0.4	2	27.85	133.29	0.00	0.022	28.44	132.70	0.00	0.001
u2-16-0.7	3	69.98	89.09	0.00	0.015	70.31	78.39	0.00	0.001
u2-20-0.1	1	6.06	232.24	1.24	0.015	17.19	199.23	1.24	0.001
u2-20-0.4	2	20.75	216.55	1.24	0.018	39.08	198.23	1.24	0.001
u2-20-0.7	3	68.34	168.26	1.24	0.018	69.57	167.03	1.24	0.001
u2-24-0.1	3	41.11	162.80	14.14	0.038	76.20	162.80	14.14	0.001
u2-24-0.4	4	96.08	142.44	14.14	0.018	102.58	142.43	14.14	0.001
u3-18-0.1	2	0.00	371.99	0.00	0.027	36.66	293.03	0.00	0.001
u3-18-0.4	1	11.27	363.13	0.00	0.024	23.87	293.03	0.00	0.001
u3-18-0.7	3	67.00	330.02	0.00	0.019	69.31	292.69	0.00	0.001
u3-24-0.1	1	7.95	343.10	12.01	0.023	62.32	343.10	12.01	0.001
u3-24-0.4	3	25.79	358.09	12.01	0.028	74.59	343.10	12.01	0.001
u3-24-0.7	3	55.52	383.32	13.45	0.043	75.40	377.91	13.46	0.001
u3-30-0.1	1	0.00	580.17	4.50	0.030	12.36	498.52	4.50	0.002
u3-30-0.4	2	28.86	545.54	4.50	0.027	69.30	498.52	4.50	0.002
u3-30-0.7	4	94.75	484.17	6.29	0.027	96.54	480.83	6.28	0.002
u3-36-0.1	3	22.96	627.52	14.80	0.028	88.43	581.79	14.80	0.052
u3-36-0.4	3	59.89	593.05	19.57	0.031	88.43	583.35	19.57	0.002
u3-36-0.7	4	119.90	549.57	19.57	0.029	144.10	525.37	19.57	0.002
u4-16-0.1	0	0.00	232.44	8.06	0.021	0.00	186.03	8.06	0.000
u4-16-0.4	0	0.00	100.68	8.06	0.025	0.00	93.03	8.06	0.000
u4-16-0.7	3	76.64	207.02	8.06	0.022	96.62	187.04	8.06	0.000
u4-24-0.1	2	6.72	1093.30	4.67	0.024	69.77	805.15	4.67	0.001
u4-24-0.4	4	44.73	1370.86	4.67	0.029	121.85	1179.12	4.67	0.001
u4-24-0.7	3	96.64	981.14	4.67	0.028	129.32	803.55	4.67	0.001
u4-32-0.1	2	6.36	470.99	9.59	0.036	94.65	437.90	9.59	0.001
u4-32-0.4	2	28.94	541.87	9.59	0.035	54.30	437.90	9.59	0.001
u4-32-0.7	4	104.13	426.54	12.26	0.035	124.71	426.54	12.25	0.001
u4-40-0.1	2	0.92	1489.94	27.25	0.035	35.00	1448.48	27.25	0.002
u4-40-0.4	4	69.17	1605.05	27.55	0.035	109.50	1443.78	27.54	0.003
u4-48-0.1	4	27.65	826.22	34.83	0.060	117.45	736.42	34.83	0.002
u5-40-0.1	2	0.55	853.07	27.28	0.042	89.24	682.98	27.29	0.002
u5-40-0.4	4	58.82	778.42	30.40	0.042	152.10	682.08	30.40	0.001
u5-50-0.1	1	0.00	1115.87	31.08	0.043	18.32	961.06	31.08	0.002
u5-50-0.4	4	60.60	1129.49	32.80	0.045	148.77	961.59	32.80	0.002
u5-50-0.7	5	155.42	983.74	37.27	0.065	177.23	976.53	37.27	0.002

evaluation procedure returns lower total vehicle waiting times than the LP, which are reduced by 19%, on average. Note that the LP is not designed to minimize the total vehicle waiting time. Considering a specific vehicle, the LP minimizes the total waiting time at locations in which the vehicle is non-empty. Instead, the vehicle can idle as long as imposed by the time windows constraints at locations in which it is empty, without affecting the objective function value. Contrarily, the route evaluation procedure tries to assign as little waiting as possible at each encountered location in the vehicle route. As such, the route evaluation procedure returns excess-ride-time optimal schedules which minimize completion time.

With respect to the total charging time, it can be noted that the battery heuristic returns higher total vehicle charging times. In particular, the charging time per vehicle is on average doubled in the battery heuristic than in the solution obtained from the LP. As such, the battery heuristic maximizes the battery SOC while maintaining scheduling feasiblity. Indeed, the LP is not designed to maximize (nor minimize) battery levels but only to provide battery-feasible solutions. As such, a solution maximizing or minimizing the total battery inventory is regarded as equivalent by the LP. Note that some of the studied instances consider visiting recharging stations before returning to the destination depot, even when recharging is not needed. In particular, this case realizes when the location of the visited charging station coincides with the location of the destination depot and the minimum battery inventory ratio is un-binding, as for example in instance u3-18-0.1, u3-30-0.1, and u5-50-0.1. Even in those specific cases, the battery heuristic returns maximal recharging times and as such vehicle battery inventories are maximized at the destination depots.

4.6 Summary

In this study, we proposed an excess-ride-time procedure for scheduling dial-a-ride instances. The procedure is shown optimal and can be potentially employed to: 1) reduce mixed-integer-linear programs by time-dependent decision variables and constraints, even when the objective function includes time-related aspects; 2) efficiently solve scheduling problems from static and dynamic metaheuristic appraches. For the e-ADARP, we further propose a battery heuristic which can be used when the vehicle routes include one or more visits to charging facilities. The heuristic is needed to provide battery-feasible charging plans for excess-time-optimal vehicle schedules. However, note that the battery heuristic is not proven to be an exact procedure and builds on the assumption that charging as much as possible as early as possible is the best strategy for battery-feasibility.

Computational experiments are carried on static e-ADARP instances extracted from real ride-shares. Experiments show that the route evaluation procedure is computationally more efficient than solving a linear program, while returning optimal scheduling decisions. Furthermore, results show that the proposed excess-time-optimal scheduling procedure minimizes completion time whereas the battery heuristic maximizes the vehicle SOC. Future efforts may focus on the comparison of the proposed route evaluation procedure against state-of-the art scheduling heuristics and on larger benchmark instances from literature.

5

Conclusion

In this concluding chapter, Section 5.1 revisits the objectives, contribution, and main findings of this thesis; Section 5.2 examines practical implications and challenges related to the direct applicability of the methods discussed herein; Section 5.3 closes the present thesis by identifying and discussing interesting directions for future research.

5.1 Main Findings

This thesis proposes mathematical optimization frameworks to model and solve autonomous ride-sharing operations. The introduced problem is denoted by the electric autonomous dial-a-ride problem, i.e. the e-ADARP, and incorporates considerations related to battery and autonomous management, other than typical dial-a-ride aspects. In particular, the autonomy of the vehicles is treated by relaxing maximum-route-duration constraints and by considering a set of optional destination depots, as autonomous vehicles can operate non-stop and relocate to optional parking locations within the city limits. Furthermore, the autonomous nature of the fleet imply vehicles can continuously detour to serve upcoming requests or according to other changing conditions (e.g. traffic congestion, among others). Battery management is modeled through decision variables and constraints which are designed to let vehicles recharge as much as needed at the visited charging facilities. As such, new features incorporated in the problem definition include charging stations, multiple destination depots, partial recharge times, and final battery level requirements. The goal of the problem is to minimize the total operational cost, measured by the vehicle total travel time, and user dissatisfaction, measured by the extra time users spend on-board because of ride-sharing.

The e-ADARP is studied in light of static scenarios, in which all demand is known in advance and needs to be served, and dynamic scenarios, in which demand is revealed on-line and may be rejected. In Chapter 2, the static e-ADARP is modeled through a 3-indexed and a 2-indexed mixed integer linear program which is solved through a branch-and-cut framework, enhanced by problem-specific valid inequalities and separation heuristics. Extensive numerical experiments are performed on benchmark instances from literature and new instances extracted from real ride-sharing data. Results show that the introduced problem features produce a new challenging DARP variant and that the proposed problem-specific valid inequalities are among the most effective ones when battery-management aspects are most important.

In Chapter 3, the dynamic e-ADARP is modeled through an event-based simulation approach and solved through a two-phase metaheuristic approach. The metaheuristic employs a machine learning-based large neighborhood search heuristic which is trained on massive historical information from previously solved problems. The construction of the labeled dataset as well as the selection of aggregate vehicle routing features is part of the proposed approach. Extensive results are performed on real dynamic ride-sharing data and show that the proposed metaheuristic is an alternative to state-of-the art approaches and that learning from past scenarios may enhance the search under circumstances. Furthermore, results show the potential of statistical learning methods to infer appropriate neighborhood moves from local search-based metaheuristics and highlight the most determinant vehicle routing features from the proposed optimization policy.

Finally, Chapter 4 proposes an exact scheduling procedure which can be employed to find excess-ride-optimal plans for the e-ADARP, as well as other dial-a-ride problems. Specifically for the e-ADARP, we provide a battery management heuristic which supplements the scheduling procedure by providing recharging plans. Results show that the proposed scheduling algorithm is a competitive alternative to a linear-program approach and that it can be used to efficiently reduce dial-a-ride formulations or in the context of on-line operations. Furthermore, results show that the proposed procedure minimizes the service completion time while maximizing battery inventory levels at the end of service.

5.2 Practical Implications

Autonomous ride-sharing is a relatively new business model which has recently been deployed in several countries, including Switzerland (CNN, 2018). According to various estimates, ride-sharing already attracts more than 50,000 Uber pools per week in New York City and is expected to further develop with the introduction of autonomous rides (Mikaela, 2015). Given that the success of ride-sharing resides on the efficiency of its services, this thesis offers methological frameworks to practically manage and quantify the benefits of autonomous fleets, with the aim of maximizing user satisfaction.

The proposed operational frameworks in this thesis build on real-world ride-sharing instances and may be equally applied to other demand scenarios. Interesting practical insights from this thesis include a comparison between the quality of service offered by autonomous and conventional vehicles. Indeed, given that empty autonomous vehicles may need to use some of their service time to recharge, the use of an autonomous fleet naturally pushes towards carpooling solutions. However, this feature does not heavily impact operational cost, which in this thesis is shown to only increase by an average of 2% between conventional and autonomous vehicles. Similarly, the consideration of level of service along with operational cost aspects allows to sensibly decrease user detours, at no more than 2% extra cost. This thesis also offers a comparison between state-of-the-art optimization policies, which are demand-agnostic, with the proposed machine learning-based optimization policy in Chapter 3, which exploits information from past ride-shares. As shown in this thesis, in certain instances, ride-sharing service may be greatly improved by making optimal decisions which are learned from past demand patterns. The quality of autonomous ride-sharing mainly depends on user satisfaction, response time, and vehicle availability. As such, Chapter 4 proposes an efficient service-oriented scheduling algorithm which minimizes service completion time and returns optimal timing plans within a few milliseconds. In the studied problem, vehicles are electric and their availability for service correlates to their state of charge. Indeed, discharged vehicles are unavailable for service and need to idle as long as to sufficiently recharge their battery inventories. As such, maximizing vehicle battery levels at visited recharging facilities is a crucial aspect for electric autonomous ride-sharing. As shown by the results from the battery management algorithm from Chapter 4, recharging phases can be appropriately planned and maximized at the visited stations, without penalizing the level of service.

5.3 Future Research Directions

This thesis treats a complex passenger mobility problem integrating autonomous mobility and ride-sharing, whose solution relies on the efficient implementation of algorithms combining concepts from optimization, simulation, and statistical learning. The problem is relatively new and thus leads towards multiple future directions and research opportunities.

From an operational perspective and in the specifics of the Swiss case, the use of autonomous ride-sharing may be further studied with respect to the opportunities it offers to *connect de-centralized villages to main city centers* (Cueni, 2016). Specifically, some of these villages need to be connected to main transportation hubs through a public transport service, which is currently unavailable due to infrastructure limitations. Given the sparse demand for transport generated from such locations, research directions may consider investigating the use of on-demand dial-a-ride transit to provide connection to other transportation services, e.g. railways, main bus lines, etc. From an optimization perspective, this direction would involve deriving optimization models which consider optional transfers and coordination with several transportation options.

In the most general context of on-line autonomous ride sharing, an interesting research direction may also focus on the modeling *look-ahead strategies* to drive optimization frameworks by estimates of future demand. Indeed, ride-sharing demand is a complex phenomenon to be modeled, let alone foresee. However, using approximate models to forecast ride-sharing demand may induce interesting operational decisions and conclusions. For example, it may be beneficial to reject some requests if this is expected to decrease the total number of served requests later on. Concurrently, it may be beneficial to have information on future ride-shares to *strategically relocate* vehicles in the serviced area and design appropriate *pricing schemes*.

Finally, the combination of *machine learning and optimization* leads to very interesting research directions. Indeed, there are several ride-sharing phenomena which can only be approximately modeled analytically and can instead be more precisely depicted by extrapolating data-driven models. This relates to real-world phenomena, as for example ride-sharing demand, but also more abstract phenomena specifically related to the way optimization algorithms operate. For example, machine learning algorithms could approximate complex optimal decisions from optimization policies. In this thesis we attempt to achieve results in this direction by trying to learn which algorithm is most efficient in explore neighboring solutions within a large neighborhood search metaheuristic. Our results suggest that there is room for improvement and further investigation in this direction.

References

Albareda-Sambola, M., Fernández, E., and Laporte, G. (2014). “The dynamic multiperiod vehicle routing problem with probabilistic information”. In: *Computers & Operations Research* 48, pp. 31–39.

Alonso-Mora, J. et al. (2017). “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment”. In: *Proceedings of the National Academy of Sciences* 114.3, pp. 462–467.

Archetti, C., Fernández, E., and Huerta-Muñoz, D. L. (2018). “A two-phase solution algorithm for the Flexible Periodic Vehicle Routing Problem”. In: *Computers & Operations Research* 99, pp. 27–37.

Arslan, O., Yıldız, B., and Karaşan, O. E. (2015). “Minimum cost path problem for plug-in hybrid electric vehicles”. In: *Transportation Research Part E* 80, pp. 123–141.

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). “A polyhedral study os the asymmetric traveling salesman problem with time windows”. In: *Networks* 36, pp. 69–79.

Attanasio, A. et al. (2004). “Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem”. In: *Parallel Computing* 30(3), pp. 377–387.

Balas, E. and Toth, P. (1983). *Branch and bound methods for the traveling salesman problem.* Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

Baldacci, R., Bartolini, E., and Mingozzi, A. (2011). “An exact algorithm for the pickup and delivery problem with time windows”. In: *Operations Research* 59(2), pp. 414–426.

Baldacci, R., Battarra, M., and Vigo, D. (2009). “Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs”. In: *Networks* 54, pp. 178–189.

Barbiroglio, E. (2020). “What Is Offsetting Improvements In Air Quality During Lockdown?” In: *Forbes*. url: `https://www.forbes.com/sites/emanuelabarbiroglio/2020/04/13/what-is-offsetting-improvements-in-air-quality-during-lockdown/#65f7759d5286` (visited on 04/13/2020).

Baugh, J. W., Kakivaya, G. K., and Stone, J. R. (1998). “Intractability of the Dial-a-Ride Problem and a multiobjective solution using simulated annealing”. In: *Engineering Optimization* 30, p. 2.

Bengio, Y., Lodi, A., and Prouvost, A. (2018). "Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon". In: *arXiv preprint arXiv:1811.06128.*

Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2012). "A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem". In: *INFORMS Journal on Computing* 24.3, pp. 343–355.

Berbeglia, G., Pesant, G., and Rousseau, L.-M. (2011). "Checking the feasibility of dial-a-ride instances using constraint programming". In: *Transportation Science* 45.3, pp. 399–412.

Bertsimas, D. and Dunn, J. (2017). "Optimal classification trees". In: *Machine Learning* 106.7, pp. 1039–1082.

Bertsimas, D., Jaillet, P., and Martin, S. (2019). "Online vehicle routing: The edge of optimization in large-scale applications". In: *Operations Research* 67.1, pp. 143–162.

Bertsimas, D. and Shioda, R. (2007). "Classification and regression via integer optimization". In: *Operations Research* 55.2, pp. 252–271.

Bonami, P., Lodi, A., and Zarpellon, G. (2018). "Learning a classification of mixed-integer quadratic programming problems". In: *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research.* Springer, pp. 595–604.

Bongiovanni, C., Kaspi, M., and Geroliminis, N. (2019). "The Electric Autonomous Dial-a-Ride Problem". In: *Transportation Research Part B: Methodological* 122, pp. 436–456.

Bongiovanni, C., Kaspi, M., and Geroliminis, N. (2020). "Scheduling Algorithm and Battery Management Heuristic for the electric Autonomous Dial-a-Ride Problem". Working paper.

Bongiovanni, C. et al. (2020). "A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic Electric Autonomous Dial-a-Ride Problem". Working paper.

Braekers, K. and Kovacs, A. A. (2016). "A multi-period dial-a-ride problem with driver consistency". In: *Transportation Research Part B: Methodological* 94, pp. 355–377.

Braekers, K., Caris, A., and Janssens, G. K. (2014). "Exact and meta-heuristics approach for a general heterogeneous dial-a-ride problem with multi depots". In: *Transportation Research Part B: Methodological* 67, pp. 166–186.

Breiman, L. (1996). "Bagging predictors". In: *Machine learning* 24.2, pp. 123–140.

Breiman, L. (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Breiman, L. (2017). *Classification and regression trees.* Routledge.

Burke, E. K. et al. (2013). "Hyper-heuristics: A survey of the state of the art". In: *Journal of the Operational Research Society* 64.12, pp. 1695–1724.

Cervero, R. (1997). *Paratransit in America: Redefining mass transportation.* Greenwood Publishing Group.

CNN (2018). *Self-driving electric bus propels Swiss town into the future.* url: `https://edition.cnn.com/2018/06/27/sport/trapeze-self-driving-autonomous-electric-bus-switzerland-spt-intl/index.html` (visited on 06/27/2018).

Cordeau, J. F. (2006). “A Branch-and-Cut Algorithm for the Dial-a-Ride Problem”. In: *Operations Research* 54(3), pp. 573–586.

Cordeau, J. F. and Laporte, G. (2003). “A tabu search heuristic for the static multi-vehicle dial-a-ride problem”. In: *Transportation Research Part B: Methodological* 37(6), pp. 579–594.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). “A unified tabu search heuristic for vehicle routing problems with time windows”. In: *Journal of the Operational research society* 52.8, pp. 928–936.

Coslovich, L., Pesenti, R., and Ukovich, W. (2006). “A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem”. In: *European Journal of Operational Research* 175(3), pp. 1605–1615.

Cueni, R. (2016). “Through the city in a driverless Postbus”. In: *Swiss Post.* url: `https://geschaeftsbericht.post.ch/15/ar/en/annual_report/business_performance/passenger_transport_market/autonomous_electric_shuttle_buses.htm` (visited on 06/23/2016).

Desaulniers, G. et al. (2016). “Exact algorithms for electric vehicle-routing problems with time windows”. In: *Operations Research* 64(6), pp. 1388–1405.

Dial, R. (1995). “Autonomous dial-a-ride transit introductory overview”. In: *Transportation Research Part C: Emerging Technologies* 3.5, pp. 261–275.

Diana, M. and Dessouky, M. M. (2004). “A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows”. In: *Transportation Research Part B: Methodological* 38(6), pp. 539–557.

Diana, M., Dessouky, M. M., and Xia, N. (2006). “A model for the fleet sizing of demand responsive transportation services with time windows”. In: *Transportation Research Part B: Methodological* 40.8, pp. 651–666.

Doppstadt, C., Koberstein, A., and Vigo, D. (2016). “The Hybrid Electric Vehicle-Traveling Salesman Problem”. In: *European Journal of Operational Research* 253, pp. 852–842.

Dunning, I., Huchette, J., and Lubin, M. (2017). “JuMP: A modeling language for mathematical optimization”. In: *SIAM Review* 59.2, pp. 295–320.

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

Erdoğan, S. and Miller-Hooks, E. (2012). “A Green Vehicle Routing Problem”. In: *Transportation Research Part E: Logistics and Transportation Review* 48(1), pp. 100–114.

Fagnant, D. J. and Kockelman, K. (2015). “Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations”. In: *Transportation Research Part A: Policy and Practice* 77, pp. 167–181.

Felipe, Á. et al. (2014). “A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges”. In: *Transportation Research Part E* 71, pp. 111–128.

Ferrucci, F. and Bock, S. (2015). “A general approach for controlling vehicle en-route diversions in dynamic vehicle routing problems”. In: *Transportation Research Part B* 77, pp. 76–87.

Ferrucci, F. and Bock, S. (2016). “Pro-active real-time routing in applications with multiple request patterns”. In: *European Journal of Operational Research* 253, pp. 356–371.

Ferrucci, F., Bock, S., and Gendreau, M. (2013). “A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods”. In: *European Journal of Operational Research* 225.1, pp. 130–141.

Funke, B., Grünert, T., and Irnich, S. (2005). “Local search for vehicle routing and scheduling problems: Review and conceptual integration”. In: *Journal of heuristics* 11.4, pp. 267–306.

Genikomasakis, K. and Mitrentsis, G. (2017). “A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications”. In: *Transportation Research Part D: Transport and Environment* 50, pp. 98–118.

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

Gomes, C. P. and Selman, B. (2001). “Algorithm portfolios”. In: *Artificial Intelligence* 126.1-2, pp. 43–62.

Greenblatt, J. B. and Saxena, S. (2015). “Autonomous taxis could greatly reduce greenhouse-gas emissions of US light-duty vehicles”. In: *Nature Climate Change* 5, pp. 860–863.

Grover, P. (2018). “5 Regression Loss Functions All Machine Learners Should Know: Choosing the right loss function for fitting a model”. In: *Medium Heartbeat.* url: `https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0` (visited on 06/05/2018).

Gschwind, T. and Drexl, M. (2019). “Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem”. In: *Transportation Science* 53.2, pp. 480–491.

Gschwind, T. and Inrich, S. (2014). “Effective handling of dynamic time windows and its application to solving the dial-a-ride problem”. In: *Transportation Science* 49(2), pp. 335–354.

Gunluk, O. et al. (2016). “Optimal generalized decision trees via integer programming”. In: *arXiv preprint arXiv:1612.03225.*

Häme, L. and Hakula, H. (2015). “A maximum cluster algorithm for checking the feasibility of dial-a-ride instances”. In: *Transportation Science* 49.2, pp. 295–310.

Healy, P. and Moll, R. (1995). “A new extension of local search applied to the Dial-A-Ride Problem”. In: *European Journal of Operations Research* 83 (1), pp. 83–104.

Hiermann, G. et al. (2016). “The electric fleet size and mix vehicle routing problem with time windows and recharging stations”. In: *European Journal of Operational Research* 252, pp. 995–1018.

Hunsaker, B. and Savelsbergh, M. (2002). “Efficient feasibility testing for dial-a-ride problems”. In: *Operations research letters* 30.3, pp. 169–173.

Hyland, M. and Mahmassani, H. S. (2020). “Operational benefits and challenges of shared-ride automated mobility-on-demand services”. In: *Transportation Research Part A: Policy and Practice* 134, pp. 251–270.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2006). “Exploiting knowledge about future demands for real-time vehicle dispatching”. In: *Transportation Science* 40(2), pp. 211–225.

Jaw, J.-J. et al. (1986). “A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows”. In: *Transportation Research Part B: Methodological* 20.3, pp. 243–257.

Kerschke, P. et al. (2019). “Automated algorithm selection: Survey and perspectives”. In: *Evolutionary computation* 27.1, pp. 3–45.

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

Kruber, M., Lübbecke, M. E., and Parmentier, A. (2017). “Learning when to use a decomposition”. In: *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems.* Springer, pp. 202–210.

Laker, L. (2020). “Milan announces ambitious scheme to reduce car use after lockdown”. In: *The Guardian.* url: `https://www.theguardian.com/world/2020/apr/21/milan-seeks-to-prevent-post-crisis-return-of-traffic-pollution` (visited on 04/21/2020).

Larsen, E. et al. (2019). “Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information”. In: *arXiv preprint arXiv:1901.07935.*

Li, B. et al. (2016). “An adaptive large neighborhood search heuristic for the share-a-ride problem”. In: *Computers & Operations Research* 66, pp. 170–180.

Liaw, A., Wiener, M., et al. (2002). “Classification and regression by randomForest”. In: *R news* 2.3, pp. 18–22.

Lodi, A. and Zarpellon, G. (2017). “On learning and branching: a survey”. In: *Top* 25.2, pp. 207–236.

Lu, Q. and Dessouky, M. M. (2004). “An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem”. In: *Transportation Science* 38(4), pp. 503–514.

Lysgaard, J. (2006). “Reachability cuts for the vehicle routing problem with time windows”. In: *European Journal of Operational Research* 175, pp. 210–223.

Malitsky, Y. et al. (2013). “Algorithm portfolios based on cost-sensitive hierarchical clustering”. In: *Twenty-Third International Joint Conference on Artificial Intelligence*.

Marković, N. et al. (2015). “Optimizing dial-a-ride services in Maryland: benefits of computerized routing and scheduling”. In: *Transportation Research Part C: Emerging Technologies* 55, pp. 156–165.

Masmoudi, M. A. et al. (2018). “The dial-a-ride problem with electric vehicles and battery swapping stations”. In: *Transportation Research Part E: Logistics and Transportation Review* 118, pp. 392–420.

Masmoudi, M. A. et al. (2016). “Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem”. In: *Transportation Research Part E: Logistics and Transportation Review* 96, pp. 60–80.

Mikaela (2015). “Who says New Yorkers don’t like to share?” In: *Uber Newroom*. url: `https://www.uber.com/newsroom/who-says-new-yorkers-dont-like-to-share/` (visited on 11/09/2015).

Molenbruch, Y., Braekers, K., and Caris, A. (2017). “Typology and literature review for dial-a-ride problems”. In: *Annals of Operations Research* 259 (1-2), pp. 295–325.

Molenbruch, Y. et al. (2017). “Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation”. In: *Computers & Operations Research* 77, pp. 58–71.

Nikolaev, A. G. and Jacobson, S. H. (2010). “Simulated annealing”. In: *Handbook of metaheuristics*. Springer, pp. 1–39.

Parragh, S. N. (2011). “Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem”. In: *Transportation Research Part C: Emerging Technologies* 19(5), pp. 912–930.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2010). “Variable neighborhood search for the dial-a-ride problem”. In: *Computers & Operations Research* 37.6, pp. 1129–1138.

Parragh, S. N. and Schmid, V. (2013). “Hybrid column generation and large neighborhood search for the dial-a-ride problem”. In: *Computers & Operations Research* 40(1), pp. 490–497.

Parragh, S. N. et al. (2009). “A heuristic two-phase solution approach for the multi-objective dial-a-ride problem”. In: *Networks: An International Journal* 54.4, pp. 227–242.

Peleda, I. et al. (2019). “Online Predictive Optimization Framework for Stochastic Demand-Responsive Transit Services”. In: *stat* 1050, p. 21.

Pelletier, S., Jabali, O., and Laporte, G. (2016). “50th Anniversary Invited Article - Goods Distribution with Electric Vehicles: Review and Research Perspectives”. In: *Transportation Science* 50(1), pp. 3–22.

Pelletier, S., Jabali, O., and Laporte, G. (2018). “Charge scheduling for electric freight vehicles”. In: *Transportation Research Part B: Methodological* 115, pp. 246–269.

Pelletier, S. et al. (2017a). “Battery degradation and behavior for electric vehicles: Review and numerical analyses of several models”. In: *Transportation Research Part B: Methodological* 103, pp. 158–187.

Pelletier, S. et al. (2017b). “Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models”. In: *Transportation Research Part B: Methodological* 103, pp. 158–187.

Perry, T. S. (2020). “Uh-oh Uber, Look Out Lyft: Here Comes Driverless Ride Sharing”. In: *IEEE Spectrum.* accessible at: https://spectrum.ieee.org/view-from-the-valley/transportation/self-driving/uhoh-uber-look-out-lyft-here-comes-driverless-ride-sharing. Accessed: March 19, 2020.

Pisinger, D. and Ropke, S. (2010). “Large neighborhood search”. In: *Handbook of metaheuristics.* Springer, pp. 399–419.

Potvin, J.-Y. and Rousseau, J.-M. (1993). “A parallel route building algorithm for the vehicle routing and scheduling problem with time windows”. In: *European Journal of Operational Research* 66.3, pp. 331–340.

Powell, W. B. et al. (2002). “Implementing real-time optimization models: A case application from the motor carrier industry”. In: *Operations Research* 50.4, pp. 571–581.

Raviv, T. and Kaspi, M. (2012). “The locomotive fleet fueling problem”. In: *Operations Research Letters* 40, pp. 39–45.

Ropke, S. and Cordeau, J. F. (2009). “Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows”. In: *Transportation Science* 43(3), pp. 267–286.

Ropke, S., Cordeau, J. F., and Laporte, G. (2007). “Models and branch-and-cut algorithms for pickup and delivery problems with time windows”. In: *Networks* 49(4), pp. 258–272.

Ropke, S. and Pisinger, D. (2006). “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows”. In: *Transportation science* 40.4, pp. 455–472.

Ruland, K. S. and Rodin, E. Y. (1997). “The pickup and delivery problem: Faces and branch-and-cut algorithm”. In: *Computers & mathematics with applications* 33(12), pp. 1–13.

Sacramento, D., Pisinger, D., and Ropke, S. (2019). “An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones”. In: *Transportation Research Part C: Emerging Technologies* 102, pp. 289–315.

Savelsbergh, M. W. (1985). “Local search in routing problems with time windows”. In: *Annals of Operations Research* 4 (1985/6), pp. 285–305.

Savelsbergh, M. W. (1992). “The vehicle routing problem with time windows: Minimizing route duration”. In: *ORSA Journal on Computing* 4, pp. 146–154.

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

Schilde, M., Doerner, K., and Hartl, R. (2011a). “Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports”. In: *Computers & Operations Research* 38(12), pp. 1719–1730.

Schilde, M., Doerner, K. F., and Hartl, R. F. (2011b). “Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports”. In: *Computers & operations research* 38.12, pp. 1719–1730.

Schilde, M., Doerner, K. F., and Hartl, R. F. (2014). “Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem”. In: *European journal of operational research* 238.1, pp. 18–30.

Schneider, M., Stenger, A., and Goeke, D. (2014). “The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations”. In: *Transportation Science* 48(4), pp. 500–520.

Shaw, P. (1997). “A new local search algorithm providing high quality solutions to vehicle routing problems”. In: *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.*

Simonetto, A., Monteil, J., and Gambella, C. (2019). “Real-time city-scale ridesharing via linear assignment problems”. In: *Transportation Research Part C: Emerging Technologies* 101, pp. 208–232.

Soper, T. (2015). “Lyft’s carpooling service now makes up 50% of rides in San Francisco; 30 % in NYC”. In: *GeekWire*. accessible at: http://www.geekwire.com/2015/lyfts-carpooling-service-now-makes-up-50-of-rides-in-san-francisco-30-in-nyc/. Accessed:February 1, 2017.

Talebpour, A. and Mahmassani, H. S. (2016). “Influence of connected and autonomous vehicles on traffic flow stability and throughput”. In: *Transportation Research Part C: Emerging Technologies* 71, pp. 143–163.

Toth, P. and Vigo, D. (2014). *Vehicle Routing Problems, Methods, and Applications (Second Edition).* SIAM, Philadelphia Monograph on Discrete Mathematics and Applications.

Tran, T. H. et al. (2018). “An efficient heuristic algorithm for the alternative-fuel station location problem”. In: *European Journal of Operational Research* 269.1, pp. 159–170.

Vigo, D. et al. (2019). “Enhancing Local Search Through Machine Learning: a Case Study on the Vehicle Routing Problem”. Workshop of the EURO Working Group on Vehicle Routing and Logistics optimization (VeRoLog). url: `https://verolog2019.sciencesconf.org/251443/document`.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). “Pointer networks”. In: *Advances in Neural Information Processing Systems*, pp. 2692–2700.

Weisstein, E. W. (2020). “Law of Cosines”. In: *MathWorld – A Wolfram Web Resource*. url: `https://mathworld.wolfram.com/LawofCosines.html` (visited on 03/23/2020).

Curriculum vitae

Claudia **Bongiovanni**

claudia.bongiovanni@epfl.ch | +41 79 661 13 48
Linkedin: Link | Google scholar: Link

EDUCATION

PhD, Civil & Environ. Engineering
EPFL
June 2020 | Lausanne, Switzerland

MSc, Civil & Environ. Engineering
University of California, Berkeley
May 2013 | Berkeley (CA), USA

BSc, Civil & Environ. Engineering
Università degli Studi Roma Tre
July 2012 | Rome, Italy

SKILLS

Hard skills

- Numerical Optimization
  [On graphs: Routing, Scheduling, Network Flow]
  (ex: MILP, LP, B&C, Metaheuristics)
- Numerical Simulation
  (ex: Event-Based, Bootstrapping)
- Statistical Analysis
  (ex: Bootstrapping, Variance Reduction)
- Machine Learning [Python/Scikit]
  (ex: Linear and Logistic Regression, Random Forests, SVM)
- Computer Programming [Serial/Parallel]
  (ex: Object-oriented, Functional)

Soft skills

- Project Management
- Scientific presentation/reporting
- Academic writing

Programming Languages

Expert: Julia • MatLab • LaTeX
Familiar: Python • Java • Scala • R

Optimization Solvers

Gurobi • CPLEX

Softwares / Editors

Jupyter • Atom • Microsoft Office Suite

PUBLICATIONS

- **Bongiovanni, C.**, Kaspi, M. & Geroliminis, N. (2019). "The electric Autonomous Dial-a-Ride Problem". ***Transportation Research Part B: Methodological*** 122, pp. 436-456.
- **Bongiovanni, C.**, Kaspi, M., Cordeau, J-F. & Geroliminis, N. (2020). "A Machine Learning-Based Two-Phase Metaheuristic for the Dynamic Electric Autonomous Dial-a-Ride Problem"(***working paper***)
- **Bongiovanni, C.**, Kaspi, M., & Geroliminis, N. (2020). "An Exact Scheduling Algorithm and Battery Inventory Heuristic for the electric Autonomous Dial-a-Ride Problem"(***working paper***)

PROFILE

Doctor in operations research with 5+ years experience in numerical optimization, numerical simulation, statistical analysis, and data science to model and solve network flow and combinatorial optimization problems for autonomous transportation.

EXPERIENCE

Research and Teaching Assistant | EPFL
September 2015 - Present | Lausanne, Switzerland

- Modeled hardly-constrained combinatorial optimization problems for electric autonomous ride-sharing
- Developed an exact solution framework (branch-and-cut) enhanced by problem-specific valid inequalities
- Developed a discrete event-based simulation framework for on-line ride-sharing
- Developed a machine learning-based metaheuristic within a two-phase heuristic framework
- Designed an efficient and optimally-proven scheduling algorithm and battery inventory heuristic procedure
- Involved in the implementation of a pilot test using autonomous vehicles in the city of Sion, CH (with: Bestmile Inc. and PostBus Ltd.)
- Involved in teaching and supervision of bachelor and master students

Visiting Doctoral Researcher | CIRRELT
April 2018 - July 2018 | Montréal, Canada

- Defined a research direction combining the fields of machine learning and optimization (in collaboration with Prof. Jean-François Cordeau)
- Analyzed operational policies for on-line ride-sharing through a simulation-based optimization approach
- Collected a 600,000+ labeled dataset and extracted significant vehicle routing problem features

Signal Processing Intern | PEER
June 2013 - August 2013 | Richmond (CA), USA

- Performed analysis (Fourier) over seismic data for the "NGA-East Project".
- Developed parsers to automatize data handling.

TEACHING

Graduate Teaching Assistant | EPFL | UC Berkeley
September 2015 - Spring 2019 | Lausanne, Switzerland
"Fundamentals of traffic operations and control ", "Global Issues: Mobility B ".
August 2014 - December 2014 | Berkeley, CA
"Introduction to computer programming for scientists and engineers ".

Supervision of Master Thesis | EPFL

"Anouk Allenspach, 2018. "The online electric autonomous dial-a-ride problem: A dynamic insertion algorithm".",

"Luc Chodron de Courcel, 2018. "The electric autonomous dial-a-ride problem".",

"Enrico Agosti, 2016. "Optimization of autonomous public transportation: applications and solutions".",

"Dai Qichun, 2016. "Optimization of public transport service with autonomous vehicles"."

ACADEMIC AWARDS

- Mobility Award, EPFL | 2017
- Best Presentation Award, hEART | 2016
- Western North America Fellow, SPE | 2014
- Jane Lewis Fellow, UC Berkeley | 2012-13
- International Fellow, U Roma Tre | 2010

REVIEWING

- Transportation Science
- Tranportation Research Part B
- Transportation Research Part E
- IEEE Transactions on ITS
- IEEE ITS Magazine
- Transportation Research Record

LANGUAGES

Italian (Mother tongue)
French (Fluent)
English (Fluent)
German (Basic)

HOBBIES

Swimming
Skiing
Figure skating
Classical piano

CONFERENCES AND SEMINARS

Bongiovanni, C., Kaspi, M., Cordeau, J.-F., Geroliminis, N. " *A learning large neighrborhood search for the dynamic electric autonomous dial-a-ride problem*".

Presented at:

- 8th hEART conference, September 2019
- 19th STRC conference, May 2019
- 7th INFORMS TSL conference, July 2019
- 7th INFORMS VeRoLog conference, June 2019.

Bongiovanni, C., Kaspi, M., Geroliminis, N. "*A two-phase heuristic approach for the dynamic electric autonomous dial-a-ride problem*".

Presented at:

- 7th hEART conference, September 2018
- The 19th EU/ME workshop on metaheuristics for industry, March 2018.

Bongiovanni, C., Kaspi, M., Geroliminis, N. "*The electric autonomous dial-a-ride problem*".

Presented at:

- CIRRELT seminar, Université de Montréal May 2018.
- 6th hEART conference, September 2017
- 6th INFORMS VeRoLog conference, July 2017
- 17th STRC conference, May 2017
- 14th Swiss OR Days, September 2016
- 28th INFORMS EURO conference, July 2016.

Bongiovanni, C., Kaspi, M., Geroliminis, N. "*Dial-a-ride transit systems: Optimization and adaptive network design strategies for large complex networks*".

Presented at:

- AIIT seminar, Politecnico di Torino, September 2016
- 5th hEART conference, September 2016
- 16th STRC conference, May 2016.