POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Exact and Heuristic Algorithms for Large-Scale Dial-a-Ride Problems: A Study of Practical and Electric Variants

MOHAMMAD KARIMI
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie industriel

Août 2025

© Mohammad Karimi, 2025.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Exact and Heuristic Algorithms for Large-Scale Dial-a-Ride Problems: A Study of Practical and Electric Variants

présentée par **Mohammad KARIMI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Antoine LEGRAIN, président
Michel GENDREAU, membre et directeur de recherche
Guy DESAULNIERS, membre et codirecteur de recherche
Jorge MENDOZA, membre
Jakob PUCHINGER, membre externe

DEDICATION

À mon amour,
présence silencieuse dans chaque battement de mon cœur,
lumière douce dans l'ombre de mes doutes,
et souffle fidèle au creux de mon âme...

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Professor Michel Gendreau, my thesis supervisor, for his insightful guidance, thoughtful advice, and continuous encouragement throughout my doctoral studies. His expertise in operations research and his strategic perspective have played a key role in shaping this work. I am especially thankful for his support in both the academic and professional aspects of my journey.

I am equally grateful to Professor Guy Desaulniers, my co-supervisor, for his invaluable contributions, rigorous feedback, and unwavering availability throughout the research process. His deep knowledge of mathematical optimization and his precise approach to problem-solving have greatly enriched the quality of this thesis.

I would also like to sincerely thank Professor Antoine Legrain, who has accepted the position of president of my thesis jury, and Professors Jorge Mendoza and Jakob Puchinger, who have kindly agreed to serve as jury members. I am honored by their participation and look forward to their feedback and insights.

I gratefully acknowledge the collaboration and support of GIRO Inc., whose involvement provided not only access to real-world data and industrial challenges, but also the opportunity to ensure that the research outcomes are practically relevant and applicable. Their trust and partnership have been instrumental to the success of this work.

Lastly, I wish to express my heartfelt gratitude to my wife, whose love, patience, and constant encouragement have been the foundation of my strength throughout this doctoral journey. Her support has made everything possible.

RÉSUMÉ

Cette thèse porte sur le problème de transport à la demande avec réservation préalable, plus connu sous le nom de Dial-a-Ride Problem (DARP), qui constitue une classe fondamentale de problèmes de tournées de véhicules. Le DARP trouve son application dans les systèmes de transport à la demande, tels que les services de transport adapté, le transport médical et les plateformes de covoiturage. Il s'agit d'organiser des itinéraires de véhicules afin de satisfaire un ensemble de demandes de transport, tout en respectant diverses contraintes opérationnelles telles que les fenêtres temporelles, les temps de trajet maximaux des passagers et les limites de capacité des véhicules. Dans les applications réelles, la complexité du DARP est exacerbée par des contraintes pratiques, la taille des instances à traiter, et l'émergence de nouvelles exigences liées à l'introduction des véhicules électriques. L'objectif principal de cette recherche est de concevoir des algorithmes d'optimisation performants capables de résoudre ces variantes réalistes et de très grande taille du DARP, en alliant rigueur dans la modélisation et efficacité en termes de temps de calcul.

La première contribution de la thèse s'intéresse à une version pratique du DARP, incluant des passagers hétérogènes (utilisateurs ambulants ou en fauteuil roulant), une flotte de véhicules diversifiée, et des règles contractuelles spécifiques. Cette version tient compte de contraintes avancées telles que la durée maximale des tournées, les pauses obligatoires selon les contrats, les budgets limités par contrat, ainsi que la compatibilité entre les passagers et les véhicules. Le problème est modélisé sous forme d'un programme linéaire en nombres entiers de type partitionnement d'ensemble, comportant un nombre exponentiel de variables. Pour le résoudre, nous avons développé un algorithme exact de type branch-price-and-cut, combinant génération de colonnes, séparation de coupes et exploration d'un arbre de recherche. Un algorithme d'étiquetage sur mesure a été conçu pour résoudre efficacement les sous-problèmes de génération de colonnes, tout en assurant la faisabilité des tournées générées. Les expérimentations réalisées sur des données réelles fournies par GIRO Inc., entreprise montréalaise spécialisée dans les logiciels de planification de transport public, montrent l'efficacité de notre approche. L'algorithme a permis de résoudre de manière optimale des instances comportant jusqu'à 849 demandes et plus de 70 véhicules hétérogènes répartis sur cinq contrats distincts, ce qui constitue, à notre connaissance, la plus grande instance de DARP jamais résolue de façon optimale dans la littérature. Ces résultats ont également permis d'évaluer les performances de l'algorithme heuristique utilisé par notre partenaire industriel.

La deuxième contribution traite du défi posé par les instances de très grande taille, pour

lesquelles les méthodes exactes ne sont plus applicables en pratique. Pour cela, nous proposons un algorithme de type Variable Neighborhood Search (VNS) conçu pour traiter les contraintes opérationnelles du DARP tout en limitant l'augmentation des temps de calcul en fonction de la taille des instances. L'algorithme repose sur une phase de génération de solution initiale hybride combinant des heuristiques de construction et une procédure basée sur la programmation linéaire. Il intègre également des composantes exactes en programmation en nombres entiers mixtes pour améliorer la recherche locale et la transition entre les voisinages. Différentes structures de voisinage (échange, chaîne, voisinages guidés par la programmation entière) sont utilisées afin d'équilibrer diversification et intensification. L'algorithme est testé sur un grand ensemble d'instances inspirées de cas réels, comportant entre 2 932 et 10 527 demandes, et des flottes allant jusqu'à 563 véhicules. Les résultats expérimentaux montrent que l'approche proposée génère systématiquement des solutions de haute qualité en moins d'une heure de calcul, surpassant les heuristiques classiques tout en produisant des solutions proches de l'optimal pour les cas de taille moyenne. Une analyse de sensibilité sur les composantes heuristiques est également présentée, apportant des recommandations concrètes pour la configuration de l'algorithme en contexte opérationnel.

La troisième contribution s'inscrit dans le contexte de la transition énergétique, en adaptant le DARP aux flottes de véhicules électriques, donnant lieu au Electric Dial-a-Ride Problem (E-DARP). Cette variante tient compte des contraintes énergétiques propres aux véhicules électriques, notamment leur autonomie limitée et la nécessité de recharge. Le problème est enrichi par plusieurs éléments réalistes qui ont souvent été omis dans la littérature : (1) une fonction de recharge concave et linéaire par morceaux reflétant le ralentissement du taux de recharge, (2) des contraintes de capacité aux stations de recharge, (3) une tarification dynamique de l'électricité selon l'heure, (4) la coexistence de différents types de bornes (rapides/lentes), et (5) la possibilité de recharges partielles selon les besoins énergétiques. Ces considérations rendent le E-DARP plus complexe que son équivalent thermique, car elles nécessitent une synchronisation fine entre la planification des tournées et la gestion énergétique.

Pour résoudre ce problème, nous étendons le cadre du VNS développé précédemment en y intégrant une stratégie d'insertion de stations de recharge, des structures de voisinage sensibles aux contraintes énergétiques, et un modèle d'optimisation pour planifier les recharges. L'algorithme est évalué sur des instances de grande taille issues de données réelles et de travaux précédents, allant jusqu'à 10 000 demandes. Les résultats montrent que notre méthode permet non seulement de produire des solutions de qualité, mais aussi d'assurer la faisabilité énergétique en tenant compte de la diversité des véhicules et des infrastructures. L'approche gère efficacement les défis opérationnels liés à la congestion aux stations, à la minimisation des coûts énergétiques, et au respect des exigences de qualité de service, dé-

montrant ainsi son potentiel pour une mise en œuvre concrète dans les systèmes de transport durable.

En résumé, cette thèse apporte trois contributions majeures à l'état de l'art en planification de tournées de véhicules dans des contextes complexes. Elle propose (1) des modèles réalistes intégrant l'hétérogénéité des usagers et des véhicules, les contraintes contractuelles et énergétiques ; (2) des algorithmes performants et évolutifs combinant exactitude et efficacité heuristique ; (3) des validations expérimentales rigoureuses démontrant l'applicabilité des méthodes développées à des cas industriels de très grande échelle. Ces travaux ouvrent des perspectives concrètes pour les agences de transport, les fournisseurs de solutions logicielles et les collectivités souhaitant améliorer l'efficacité, la durabilité et la réactivité de leurs services de transport à la demande.

ABSTRACT

This dissertation focuses on the Dial-a-Ride Problem (DARP), a fundamental class of vehicle routing problems that arises in demand-responsive transportation systems such as paratransit services, medical transport, and ride-sharing platforms. The DARP involves planning vehicle routes to fulfill transportation requests defined by pickup and delivery locations, subject to a variety of operational constraints including time windows, maximum ride times, and vehicle capacity limits. In real-world applications, the complexity of DARP increases significantly due to practical constraints, large-scale challenges, and emerging requirements related to electric vehicle operations. The overarching objective of this research is to develop high-performance optimization algorithms capable of addressing these complex and large-scale DARP variants with practical realism and computational scalability.

The first contribution of the dissertation addresses a practical DARP variant characterized by heterogeneous passengers (ambulant and wheelchair users), a diverse vehicle fleet, and contract-based operating rules. This version includes advanced constraints such as route duration limits, mandatory break patterns, budget restrictions per contract, and vehicle-resource compatibility. The problem is modeled as a set-partitioning integer linear program with an exponential number of variables. To solve it, we develop an exact branch-price-and-cut algorithm that integrates a column generation framework with branch-and-bound and cutting plane strategies. A dedicated labeling algorithm is designed to solve the pricing subproblem efficiently, ensuring route feasibility with respect to all constraints. Extensive computational experiments on real-world instances provided by GIRO Inc., a transportation software company based in Montreal, demonstrate the effectiveness of the proposed method. The algorithm successfully solves to optimality instances involving up to 849 transportation requests and more than 70 heterogeneous vehicles operating under five distinct contracts. This represents the largest DARP instance reported in the literature to be solved exactly using an exact approach. These results also enabled a critical assessment of the heuristic used by the industrial partner, thereby offering valuable feedback for operational improvement.

The second contribution targets the challenge of solving very large-scale instances of DARP, which are beyond the practical reach of exact methods due to computational limitations. We propose a variable neighborhood search (VNS) algorithm tailored for high scalability and operational realism. The algorithm incorporates a hybrid initial solution generation mechanism, combining constructive heuristics with a linear programming-based procedure. Furthermore, the VNS framework is enhanced with mixed-integer programming-based components that are

applied selectively to refine local search and neighborhood transitions. A variety of shaking strategies, including Swap, Chain, and IP-based neighborhoods, are integrated to balance intensification and diversification. The algorithm is validated on a diverse set of large- and very large-scale real-world-inspired instances, ranging from 2,932 to 10,527 transportation requests, with vehicle fleets of up to 563 units. Results indicate that the proposed method consistently generates high-quality solutions within practical time limits—often under one hour—while outperforming traditional heuristics and providing near-optimal solutions for medium-sized instances. The study also includes a sensitivity analysis of different heuristic components, offering guidance on algorithm configuration for operational planners.

The third contribution addresses the transition to electric vehicle fleets, introducing the electric dial-a-ride problem (E-DARP). In this setting, electric vehicles must service transportation requests while adhering to energy-related constraints, including limited battery capacity and recharging requirements. The problem is modeled to reflect several realistic and overlooked features, including: (1) a concave piecewise linear charging function representing the decreasing rate of charge over time, (2) station capacity constraints limiting the number of vehicles that can charge simultaneously, (3) time-dependent electricity pricing that influences charging decisions, (4) multiple charger types (e.g., fast and slow), and (5) the ability to partially charge based on route planning and energy needs. These elements make the E-DARP significantly more complex than its fuel-based counterpart, as it requires careful synchronization of routing and energy management decisions.

To solve the E-DARP, we extend the VNS framework introduced in the second contribution by embedding a charging station insertion mechanism, adaptive energy-aware neighborhood structures, and an optimization model for optimizing charging plans. The algorithm is tested on large-scale instances derived from real-world and benchmark datasets, including instances with up to 10,000 transportation requests. Results show that the proposed approach not only yields competitive routing solutions but also ensures energy feasibility across diverse vehicle types and station configurations. The method handles operational challenges such as avoiding congestion at charging stations, minimizing energy costs, and satisfying service quality constraints, demonstrating strong potential for supporting the practical deployment of electric mobility systems.

Overall, this dissertation makes three principal contributions to the state of the art in vehicle routing and transportation planning. First, it advances the modeling of DARP by integrating heterogeneous, contract-based, and energy-related constraints. Second, it introduces scalable and effective algorithmic frameworks that bridge the gap between exact methods and heuristics for solving very large and realistic instances. Third, it provides empirical ev-

idence, through rigorous computational experiments, of the applicability of these methods to real-world systems. The findings of this research have practical implications for software vendors, transit agencies, and urban planners seeking to improve the efficiency, sustainability, and responsiveness of on-demand transportation services.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT viii

LIST OF TABLES xiv

LIST OF FIGURES xvi

LIST OF SYMBOLS AND ACRONYMS xvii

LIST OF APPENDICES xviii

CHAPTER 1 INTRODUCTION 1
1.1 Context and Basic Concepts 1
1.2 Elements of the Problem 3
1.3 Research Objectives 6

CHAPTER 2 LITERATURE REVIEW 8
2.1 Dial-a-Ride Problem 8
2.2 Large-scale Routing Problem 13
2.3 Electric Dial-a-Ride Problem (E-DARP) 16

CHAPTER 3 THESIS ORGANIZATION 21

CHAPTER 4 ARTICLE 1: AN EXACT BRANCH-AND-PRICE-AND-CUT ALGORITHM FOR A PRACTICAL AND LARGE-SCALE DIAL-A-RIDE PROBLEM 23
4.1 Introduction 23
4.2 Literature review 24
4.3 Problem definition 26
4.4 Mathematical model 29
4.5 The proposed BPC algorithm 29
4.5.1 Labeling algorithm 30
4.5.2 Valid inequalities 37
4.5.3 Branching strategy 39

4.6 Computational results 39
4.6.1 Instance description 39
4.6.2 Linear relaxation results 41
4.6.3 Integer solution results 42
4.7 Conclusion 45

CHAPTER 5 ARTICLE 2: A VARIABLE NEIGHBORHOOD SEARCH ALGORITHM FOR A VERY LARGE-SCALE PRACTICAL DIAL-A-RIDE PROBLEM 48
5.1 Introduction 48
5.2 Literature review 49
5.3 Problem definition 51
5.4 The proposed algorithm 54
5.4.1 Generating the initial solution 56
5.4.2 Neighborhood classes 58
5.4.3 Local search 61
5.4.4 Break insertion 62
5.4.5 Acceptance criterion 62
5.5 Computational results 62
5.5.1 Instance description 63
5.5.2 Result of very large-scale instances 64
5.5.3 Result of medium-size instances 69
5.6 Conclusion 69

CHAPTER 6 ARTICLE 3: A VARIABLE NEIGHBORHOOD SEARCH ALGORITHM FOR THE ELECTRIC DIAL-A-RIDE PROBLEM WITH REALISTIC CHARGING CONSTRAINTS 72
6.1 Introduction 72
6.2 Literature review 73
6.3 Problem definition 75
6.4 The proposed algorithm 76
6.4.1 Generating the initial solution 77
6.4.2 Neighborhood classes 78
6.4.3 Local search 82
6.4.4 Charging station insertion 82
6.4.5 Acceptance criterion 88
6.5 Computational results 88
6.5.1 Instance description 89

6.5.2 Base Case - linear charging function 89
6.5.3 Case 2 - concave piecewise linear charging function 92
6.5.4 Case 3 - multiple charging infrastructure types 93
6.5.5 Case 4 - time-of-use electricity pricing 95
6.5.6 Case 5 - charging station capacity 96
6.6 Conclusion 100

CHAPTER 7 GENERAL DISCUSSION 101

CHAPTER 8 CONCLUSION 103

REFERENCES 105

APPENDICES 116

LIST OF TABLES

Table 2.1 Summary of large-scale routing problem studies 16
Table 2.2 Overview of recent contributions to the E-DARP 20
Table 4.1 Instance specifications 41
Table 4.2 Contract specifications 41
Table 4.3 Vehicle specifications 42
Table 4.4 Break pattern specifications and maximum route durations 42
Table 4.5 Linear relaxation results 45
Table 4.6 Integer solution results 46
Table 5.1 Instance specidfications 64
Table 5.2 Contract specifications 64
Table 5.3 Vehicle specifications 65
Table 5.4 Break pattern specifications and maximum route durations 65
Table 5.5 Performance of initial solution generation methods 67
Table 5.6 Overall performance of the shakers sequence for solving Instance S3 . 67
Table 5.7 Performance of the proposed algorithm with different initial solution generation methods 68
Table 5.8 The impact of removing family shakers on algorithm performance for solving Instance S3 70
Table 5.9 Comparative results on medium-sized instances 70
Table 6.1 Results on Cordeau instances with $\gamma = 0.1$ 90
Table 6.2 Results on Cordeau instances with $\gamma = 0.4$ 90
Table 6.3 Results on Cordeau instances with $\gamma = 0.7$ 91
Table 6.4 Results on Limmer large instances with $\gamma = 0.7$ 91
Table 6.5 Results on Limmer large instances with concave piecewise linear charging function ($\gamma = 0.7$) 92
Table 6.6 Results on Dong et al. large and very large instances with concave piecewise linear charging function ($\gamma = 0.4$) 93
Table 6.7 Results on Limmer large instances for Case 3 ($\gamma = 0.7$) 94
Table 6.8 Results on Dong et al. large and very large instances for Case 3 ($\gamma = 0.4$) 95
Table 6.9 Influence of TOU electricity pricing policy on costs - Limmer Instances ($\gamma = 0.7$) 96
Table 6.10 Comparison of the proposed algorithm with Dong et al. [1] under TOU-based electricity pricing ($\gamma = 0.4$) 98

Table 6.11 Results on Limmer large instances for Case 5 ($\gamma = 0.7$) 99
Table 6.12 Results on Dong et al. large and very large instances for Case 5 ($\gamma = 0.4$) 99

LIST OF FIGURES

Figure 1.1 A feasible route in the E-DARP 6
Figure 4.1 Distribution of the requests' start times 43
Figure 4.2 Pickup and delivery locations of the requests 44
Figure 4.3 Total cost and cost per contract for both solutions 46
Figure 4.4 Number of vehicles used per contract in both solutions 47
Figure 4.5 Number of requests covered by contract in both solutions 47
Figure 5.1 Example of a route with breaks 54
Figure 6.1 The concave piecewise linear charging functions 77
Figure 6.2 The nearest charging station time window 84
Figure 6.3 The components of the insertion model with concave piecewise linear charging functions 87
Figure 6.4 Charging process for linear and two types of concave piecewise linear charging functions 94
Figure 6.5 TOU electricity pricing policy 97

LIST OF SYMBOLS AND ACRONYMS

BB	Branch-and-bound
BC	Branch-and-cut
BP	Branch-and-price
BPC	Branch-price-and-cut
CG	Column Generation
DARP	Dial-a-Ride Problem
E-ADARP	Electric Autonomous Dial-a-Ride Problem
E-DARP	Electric Dial-a-Ride Problem
ESPPRC	Elementary Shortest Path Problem with Resource Constraints
EV	Electric Vehicle
GA	Genetic Algorithm
LNS	Large Neighborhood Search
MIP	Mixed-Integer Programming
PDP	Pickup and Delivery Problem
PDPTW	Pickup and Delivery Problem with Time Windows
RCI	Rounded Capacity Inequality
RMP	Restricted Master Problem
SoC	State-of-Charge
SP	Subproblem
SRI	Subset Row Inequality
TOU	Time-of-Use
TS	Tabu Search
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPSPD	Vehicle Routing Problem with Simultaneous Pickup and Delivery
VRPMPD	Vehicle Routing Problem with Mixed Pickup and Delivery

LIST OF APPENDICES

Appendix A Heterogeneous Dial-a-Ride Formulation 116

CHAPTER 1 INTRODUCTION

1.1 Context and Basic Concepts

The dial-a-ride problem (DARP) is a prominent class of vehicle routing problems that focuses on the efficient planning of transportation services for passengers with individual pickup and delivery requests. Formally introduced by Wilson et al. [2], the DARP is defined on a directed graph where each transportation request involves moving one or more passengers from a designated pickup location to a corresponding delivery location within a specific time window [3]. The central objective is to assign requests to a fleet of vehicles and construct their routes in a way that minimizes total operational costs—often measured in terms of total distance, total duration, or a combination of both—while respecting a variety of service and operational constraints.

At its core, the DARP generalizes the pickup and delivery problem with time windows (PDPTW) by incorporating user-centric service considerations that are especially relevant in mobility-focused applications [4]. The DARP is widely implemented in services such as paratransit transportation for elderly and disabled individuals, non-emergency medical transport, employee shuttle operations, and emerging mobility-on-demand systems [5–7]. These applications require a high level of service reliability and user satisfaction, which are often evaluated through performance indicators such as punctuality and adherence to maximum allowable ride durations. Consequently, the DARP not only emphasizes cost minimization but also the delivery of equitable and high-quality service.

Among the most common constraints found in DARP formulations are time windows, precedence, ride time limits, vehicle capacity, and route duration. Time window constraints require that each pair of pickup and delivery occurs within a predefined service interval, representing the earliest and latest permissible start times. Precedence constraints ensure that for any given request, the pickup must occur before the delivery on the same vehicle route. The ride time limit stipulates that the total duration a passenger spends in the vehicle must not exceed a specified maximum, thereby preserving comfort and limiting detours [8]. Capacity constraints reflect the limited seating or resource availability of each vehicle, meaning the total number of onboard passengers or occupied resources must never exceed the vehicle's capabilities at any point during the route. Finally, route duration constraints impose a maximum allowable operating time for each vehicle, often aligned with labor or contractual regulations in full-day scheduling contexts.

Despite its conceptual simplicity, the DARP is computationally hard to solve. Even simplified versions fall into the NP-hard class, as shown by Savelsbergh & Sol [9]. The difficulty increases significantly with the addition of realistic features such as user and vehicle heterogeneity, time-dependent travel times, dynamic request arrivals, or resource-based vehicle compatibility. As a result, exact algorithms (e.g., branch-and-price, branch-and-cut, and branch-and-price-and-cut) are effective only for small- to medium-sized instances, while heuristics and metaheuristics (e.g., tabu search, variable neighborhood search, and large neighborhood search) are typically employed for large-scale applications.

Over the years, numerous variants of the DARP have been proposed to reflect operational needs. One such variant is the multi-depot DARP [10], where vehicles can start and end their routes at different depots. Another is the dynamic DARP [11], where requests arrive in real time and must be accommodated dynamically as operations unfold. The heterogeneous DARP models passenger and vehicle differences with compatibility constraints limiting which vehicles can serve users [12, 13]. In time-dependent DARP, travel times vary depending on the time of day due to congestion or traffic conditions, affecting both route planning and feasibility [14].

The emergence of the electric dial-a-ride problem (E-DARP) represents a major evolution in the modeling of demand-responsive transportation systems [15]. In the E-DARP, vehicles are electric and have limited battery capacity. They consume energy as they travel and may require recharging at designated stations during the planning horizon. Unlike traditional fuel-based vehicles, electric vehicles (EVs) exhibit nonlinear charging behavior: charging tends to be faster when the state-of-charge of the battery is low and gradually slows down as it approaches full capacity. Consequently, recharging operations are more accurately modeled using concave piecewise linear charging functions. Moreover, charging station capacity constraints—reflecting a limited number of available charging ports—and partial recharging strategies are important to reduce service interruptions and improve system efficiency. Some advanced models also account for time-of-use electricity pricing, where the cost of charging varies throughout the day.

However, despite these practical considerations, many of these features—particularly station capacity limitations, nonlinear charging dynamics, and heterogeneous charging infrastructure—remain largely overlooked in the current E-DARP literature. This gap motivates the development of more realistic models and solution approaches capable of addressing the operational complexities inherent to electric mobility systems.

In summary, the DARP stands at the intersection of theoretical complexity and practical impact. Its rich structure and adaptability to real-world constraints make it a central problem

in transportation science. The evolution from basic formulations to highly realistic and large-scale models—including electrification, heterogeneous user needs, and contractual fleet structures—demands advanced and scalable algorithmic solutions. The remainder of this dissertation is dedicated to addressing these challenges through the development of exact and heuristic methods tailored to the demands of modern DARP applications.

1.2 Elements of the Problem

The practical DARP addressed in this dissertation is motivated by a real-world case study provided by GIRO Inc., a Montreal-based company specializing in advanced software for public and on-demand transportation systems. The problem setting reflects the operational realities faced by transportation agencies that must deliver paratransit services to a large number of users with varying mobility needs. These services must be planned within a rigid regulatory framework and under a multitude of cost, resource, and quality-of-service constraints. Unlike idealized or academic DARP formulations, the problem investigated here integrates a broad spectrum of real-world features that significantly increase the modeling complexity and computational demands. This section describes these elements in detail.

A central characteristic of the problem is the contractual structure that governs vehicle operations. The transportation agency operates according to several contracts, each associated with a specific provider, type of vehicle, and set of operational rules. For instance, one contract may involve small ambulant-accessible minivans, while another provides larger vehicles equipped to handle multiple wheelchairs. Each contract specifies a maximum number of available vehicles, a budget cap, and a cost structure based on time, distance, and usage. Furthermore, different contracts impose distinct break regulations, such as requiring mandatory rest periods if a route exceeds a certain threshold or limiting the total time a vehicle may be in service per day. These elements reflect labor laws, health and safety regulations, and economic agreements between agencies and transportation providers. Consequently, the planning process must not only satisfy operational constraints but also optimize within contract-specific limitations and cost structures.

Another key dimension of complexity arises from user heterogeneity. Passengers differ in their mobility profiles—some are ambulant and can be transported in regular seats, while others require a wheelchair and a dedicated securement space. A passenger's profile affects not only the capacity consumed onboard the vehicle but also the service duration, accessibility requirements, and vehicle compatibility. For example, not all vehicles are equipped to transport wheelchair users, and not all pickup locations are accessible by every type of vehicle. To account for this, the model incorporates a multi-resource capacity framework, where

vehicles are defined by their capacity in terms of both seats and wheelchair spaces. Each request thus involves consuming a vector of resources that must be respected throughout the route. These compatibility and capacity constraints are crucial for generating feasible and legally compliant schedules, especially in high-volume, high-heterogeneity instances.

In this context, a vehicle route is modeled as a full-day block of activities, beginning with a departure from a depot (pull-out), followed by a sequence of pickups and drop-offs, and ending with a return to the depot (pull-in). The route may include deadhead segments, during which the vehicle travels without passengers, as well as scheduled breaks. These breaks are dictated by the specific contract associated with the vehicle and must occur after certain durations of work. Importantly, a break may only be scheduled when the vehicle is empty (i.e., all passengers have been dropped off), which introduces further synchronization constraints between operational rules and routing feasibility.

To capture the operational efficiency and contractual penalties, the cost of a route is computed using a detailed multi-component cost structure. This includes fixed penalties for the number of route blocks (segments between empty states), variable costs based on time and distance traveled, and optional charges for deadhead travel or vehicle pull-outs and pull-ins. For example, routes with multiple blocks are penalized to encourage vehicle productivity by minimizing idle times. On the other hand, travel time with onboard passengers is weighted more heavily than deadhead travel to reflect its direct contribution to service delivery. The cost functions are also contract-dependent: the same vehicle performing the same route under two different contracts may yield different total costs due to distinct pricing models. This cost model is both highly detailed and non-uniform.

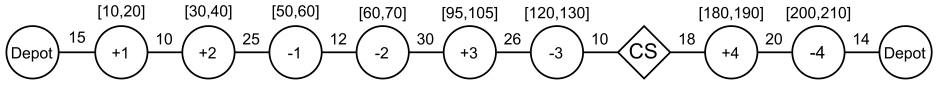
The scale of the problem further distinguishes it from traditional DARP studies. While most academic benchmarks focus on medium-sized instances, the practical datasets considered in this work include up to more than 10,000 transportation requests and more than 560 vehicles distributed across multiple contracts. Each request consists of a pickup and delivery pair with a tightly constrained time window and a maximum allowable ride time. The combination of massive request volumes, high heterogeneity, and strict feasibility requirements results in an exceptionally large and complex solution space. Classical optimization approaches—particularly those relying on explicit enumeration of feasible schedules or complete static graph formulations—become rapidly intractable at this scale.

An additional layer of complexity arises in the EV variant of the problem, commonly referred to as the E-DARP. In this extension, all vehicles are electric and operate under finite battery capacity. Each segment of a route depletes the vehicle's battery according to its distance and payload. To ensure feasibility, routes may include recharging activities at designated charging

stations. Unlike classical refueling, battery charging is governed by concave piecewise linear functions, which reflect the nonlinear behavior of real-world charging systems—vehicles charge faster at low state-of-charge and progressively slower as they approach full capacity. Furthermore, partial charging is often optimal, especially when energy is needed quickly or when station capacity is limited.

Figure 1.1 provides an illustrative example of a feasible route in the E-DARP, highlighting how vehicle load, energy consumption, and partial recharging are managed over time to satisfy operational constraints. The route includes a sequence of pickup and delivery locations, each associated with a time window, where "+x" denotes the pickup node for request x and "–x" its corresponding delivery node, and the travel time between nodes is shown along the arcs. The vehicle begins its tour from the depot with a fully charged battery and zero load. Throughout the route, the vehicle performs passenger pickups and deliveries, respecting both time window constraints and capacity limitations. The service time at each pickup and delivery node is assumed to be 1 minute, representing the time required for boarding or alighting. The lower section of the figure displays the vehicle's load, state-of-charge (SoC), and the start of service time at each node. For instance, the start of service at node "+2" occurs at time 30, and at its corresponding delivery node "–2" at time 69, resulting in a ride time of 39 minutes for request 2. Energy consumption is governed by a constant discharging rate of 0.055 kWh per kilometer, while recharging is permitted at designated charging stations at a rate of 0.11 kWh per minute. The figure illustrates a case where *partial recharging* is utilized during the route: the vehicle stops at a charging station (CS) after servicing node "–3", recharging for 30 minutes to replenish its battery sufficiently and continue serving requests. The route concludes at the depot with a SoC exactly at the minimum required threshold, set to 40% of the total battery capacity. This example highlights the importance of jointly considering routing, time scheduling, passenger constraints, and battery management—including partial recharging strategies—for the operational feasibility of electric mobility services in dial-a-ride systems.

In addition, charging operations must also respect time-of-use electricity pricing, where the cost of charging varies throughout the day. This introduces a trade-off between timing, duration, and cost of energy replenishment. Additionally, charging station capacities are finite: at any given time, only a limited number of vehicles can charge at each station. The routing algorithm must therefore synchronize vehicle arrivals and charging durations across the entire fleet while avoiding congestion at charging sites. These constraints require the integration of energy-aware scheduling and charging station insertion logic into the routing framework.



Vehicle load	0	1	2	1	0	2	0	0	2	0	0
State-of-charge	11	10.175	9.625	8.25	7.59	5.94	4.51	7.26	6.27	5.17	4.4
Start of service	4	19	30	56	69	100	127	138	180	201	216

Figure 1.1 A feasible route in the E-DARP

Overall, the problem investigated in this dissertation combines a high-fidelity representation of dial-a-ride operations, including contractual and regulatory constraints, with the computational demands of large-scale optimization and the modeling of realistic electric mobility systems. Solving this problem requires algorithms that are both structurally aware and computationally scalable, capable of navigating large search spaces while accounting for feasibility and cost at a fine level of detail. To address these needs, this research proposes a combination of exact and heuristic methods tailored to different scales and variants of the problem. The next section outlines the research objectives and methodological strategies employed in the remainder of this dissertation.

1.3 Research Objectives

The overarching objective of this dissertation is to develop and evaluate advanced optimization approaches for solving large-scale and realistic variants of the DARP, including those involving electric vehicles. The research is motivated by real-world operational challenges, particularly those faced by demand-responsive transportation systems operating under contractual, resource, and environmental constraints. Emphasis is placed on scalability, solution quality, and practical applicability. Three core research objectives guide the development of this work. First, the dissertation aims to address the limitations of existing exact methods by proposing a branch-and-price-and-cut (BPC) framework capable of solving the largest known practical instances of the DARP to proven optimality. This objective focuses on capturing key operational features such as user and vehicle heterogeneity, contract-based constraints, and time- and resource-related limitations, thereby aligning model assumptions with real-world practice.

Second, the research seeks to develop a solution approach suitable for very large-scale instances of the DARP. This objective involves the design of an algorithmic framework that maintains computational efficiency while accommodating the complexity of large transportation networks and service requirements. It emphasizes the ability to generate high-quality

solutions in reasonable computational times, making it suitable for practical planning applications.

Third, the dissertation extends the scope of DARP to electric vehicle-based transportation systems. The research objective here is to incorporate realistic features of electric mobility, such as battery dynamics, recharging constraints, infrastructure limitations, and energy cost variability, into the problem formulation and solution methodology. Of particular importance is the ability to handle very large-scale instances of the E-DARP, involving thousands of requests and diverse charging configurations. The goal is to provide decision support tools that are not only energy-aware and environmentally responsible but also scalable enough to address the operational realities of future electric fleet deployments.

Together, these objectives contribute to the development of robust and flexible optimization tools that bridge the gap between theoretical modeling and the operational needs of modern mobility systems. The approaches proposed in this dissertation are intended not only to advance the academic literature on DARP and electric vehicle routing but also to inform the design and implementation of decision support systems in real-world transport planning environments.

CHAPTER 2 LITERATURE REVIEW

This dissertation focuses on the resolution of realistic and large-scale DARP, including its electric vehicle variant (E-DARP). Section 2.1 presents the state-of-the-art in solution methods for DARP. Section 2.2 discusses algorithmic strategies developed for very large-scale routing problems, with a focus on scalability. Finally, Section 2.3 reviews the recent literature on the E-DARP, highlighting modeling advancements and optimization methods tailored to energy constraints and electric vehicle operations.

2.1 Dial-a-Ride Problem

The DARP addresses the challenge of assigning a set of user-defined transportation requests to a fleet of vehicles while generating efficient and feasible vehicle routes. It lies at the intersection of combinatorial optimization and transportation logistics and serves as a fundamental framework for modeling shared mobility systems such as paratransit services, medical transport, and emerging mobility-on-demand applications [3]. The most commonly pursued objective in DARP studies is the minimization of the service provider's operational costs, which can encompass total travel time, distance traveled, vehicle utilization, and driver working hours [16]. Alongside these, customer-centric objectives such as minimizing passenger ride time, waiting time, and deviations from desired pickup and delivery windows are also frequently addressed [17–19]. Although less prominent, environmental sustainability considerations, particularly related to vehicle emissions, have been explored in a subset of studies [20].

In addition to these general goals, various problem-specific objectives have been investigated. These include maximizing vehicle occupancy rates to improve system efficiency [21], enhancing the profitability of transportation operators [22], reducing operational burden on staff and drivers [23], and increasing the overall reliability and robustness of the transport service [24]. Such objectives underscore the multifaceted nature of the DARP, where both economic efficiency and service quality must be balanced. Furthermore, recent works have begun to incorporate social and regulatory concerns, such as equitable service provision to mobility-impaired individuals, compliance with labor laws regarding driver working hours and breaks [13], and the integration of DARP systems with other mobility platforms in the context of Mobility-as-a-Service.

While a large number of DARPs in the literature focus on a single goal, others take into

account multiple goals and require the decision-maker to choose the best answer among them. In the case of research that has considered several objective functions, it can be divided into three categories based on the method used to manage these objectives. The first type is to treat the multiple objectives as a weighted sum of different measures [6,25–27]. This method, however, is ineffective in situations when the relative relevance of each objective is uncertain or unquantifiable. In addition, the solutions are quite sensitive to the goal weights.

The second type considers goal functions hierarchically and in order of importance, where a higher-level goal must be optimized first and a lower-level goal further optimized if possible [28, 29]. This approach applies to problems where one goal is significantly more important than the others, but the relative importance of different measures cannot be demonstrated by one unit. The purpose of the third type is to obtain the Pareto frontier for the problem. The Pareto frontier contains solutions that are not dominated by any other solution in terms of relevant criteria. In addition, a complete set of non-dominated solutions can be created for the decision maker to select the appropriate program for the final implementation [16,30]. The Pareto solution approach gives the decision maker a complete picture of all possible optimal solutions, which is especially desirable when there is no confidence in the relative importance of each criterion.

Before surveying the solution algorithms proposed to solve the DARP and its many variants, it is important to highlight the range of problem features that have been considered in the literature. Algorithmically-focused studies, such as those by Parragh et al. [31, 32], Chassaing et al. [33], and Ritzinger et al. [11], typically examine a classical setting that assumes a homogeneous fleet of capacitated vehicles, strict pickup and delivery time windows, bounded route duration, and ride time limits for passengers. However, many practical applications have motivated the exploration of more complex and realistic problem characteristics. These include the presence of heterogeneous passengers and vehicle fleets [10, 12], the allowance for passenger transfers between vehicles [34–36], the integration of fixed-route and scheduled public transportation into the user journey [37–39], the inclusion of additional human resources such as attendants or medical staff [13, 23, 40], and driver scheduling requirements [19, 41]. Moreover, the recent emergence of electric vehicle routing has led to new DARP variants that account for battery capacities, charging constraints, and the positioning of charging infrastructure [15, 42, 43].

Based on the time of decision making (static or dynamic) and nature of information (deterministic or stochastic), the DARP can be classified into four categories: static-deterministic, static-stochastic, dynamic-deterministic, and dynamic-stochastic. Most of the articles consider static and deterministic DARPs, for which the decision maker has perfect information

concerning all current and future operations from the beginning of planning. Among these articles are the typical configurations of a homogeneous fleet of vehicles, time windows, maximum passenger ride time, maximum route length, and vehicle capacity. This setting is especially popular among articles that focus on algorithmic development [11, 31–33]. Except for studies focusing on algorithmic advances, most of the remaining articles focus on modeling issues related to new problem characteristics motivated by new DARP applications or actual needs in practice.

Compared to the other three categories, the number of studies in the static-stochastic category (the decision maker must determine all decisions at the beginning based on imperfect and uncertain information) seems relatively insufficient. The stochastic user arrival is considered, while the number of user requests follows a Poisson process [44]. A specific user requests services with a certain probability [45], or the arrival time of users to enter pickup points is stochastic [46]. The common goal of this research is to optimize the expectation of the objective function in anticipation of events [45, 46] or to examine the performance of the system in a static and stochastic environment [44].

Experimental research on dynamic and deterministic DARPs is usually characterized by the presentation of a simulation or other dynamic model in which decisions made in response to new information are returned to the model to influence the future evolution of the system [47]. In this version of DARPs, at any time from the beginning of planning onwards, the decision maker has perfect information concerning all current and future operations except for the appearance of new users and cancellations of users. Most research models in this category consider new user requests as events that begin the re-planning process [48, 49]. The purpose of this type of DARP is to determine how a new request will be accepted, or to make pricing decisions for new user requests [50–52]. Most of the research in this category is limited to new user requests for dynamic environments; however, in fact, there may be other types of events that may cause the transportation plan to be reconsidered, such as a vehicle breakdown and an unexpected rest break. Liang et al. [53] proposed a non-linear integer programming model to study the DARP with automated vehicles (taxis) under dynamic travel times, which is defined as a function of traffic flow. During the trip service, there is a possibility of disruption that leads to change, modification, or even cancellation of the service by users. To manage these disruptions, a recovery management framework has been proposed by Paquay et al. [54] to take care of as many patients as possible in a real-time environment. Recently, a bi-objective (transportation cost and user inconvenience) mathematical optimization model for a real-world application of the heterogeneous dynamic DARP with no request rejection has been developed by Souza et al. [55] with in-advance and urgent transportation requests.

In the category of dynamic-stochastic DARPs, various types of uncertainties are considered, such as future user requests [28, 56], random travel times [29, 56], and desired dropped-off times [57]. Stochastic information can be used to predict scenarios that may occur in the future for optimal control [58, 59]. Using random information about unknown future events to make decisions in response to recently realized events (also known as the non-myopic approach [60]), solutions are expected to be of higher quality than solutions generated by myopic methods [50–52].

To solve these increasingly complex problem settings, a wide variety of algorithms (exact, heuristic/metaheuristic, and hybrid) have been developed. Among exact algorithms, branch-and-bound (BB) frameworks remain foundational. The first branch-and-cut (BC) method for DARP was introduced by Cordeau [61], who also proposed several families of valid inequalities derived from those used in the traveling salesman problem and classical vehicle routing problems. Later studies extended this work by introducing additional classes of cuts tailored to problem-specific features, including constraints related to trip count and rest breaks [41], symmetry breaking, and driver scheduling consistency (Braekers and Kovacs, 2016), as well as extensions to electric and autonomous vehicle variants [15]. Passenger-transfer versions of DARP have been solved using BC methods enhanced by combinatorial Benders cuts that ensure feasibility across transfer points [34]. More recent efforts have introduced compact formulations that handle pairing, precedence, and capacity constraints implicitly. For instance, Rist and Forbes [62] proposed a flow-based model defined over route fragments (i.e., sequences of pickups and deliveries starting and ending with an empty vehicle), Gaul et al. [63] developed a node-based event network formulation where each vertex encodes the vehicle's onboard passenger list at each event, while Schulz and Pfeiffer [64] presented BC algorithm for DARP with incompatible customer types.

Branch-and-price (BP) algorithms address DARP by decomposing the problem into a master problem—responsible for route selection—and a pricing subproblem that generates new feasible routes. The pricing subproblem is typically modeled as an elementary shortest path problem with resource constraints and solved using labeling algorithms or dynamic programming [21]. To improve computational performance, heuristic pricing solvers may be employed first, with exact algorithms invoked only as needed [22]. A notable development in this line is the introduction of branch-price-and-cut (BPC) algorithms, which merge BP with the use of valid inequalities to tighten relaxations. Qu and Bard [65] demonstrated the benefit of subset-row inequalities in reducing BB nodes and improving convergence. Similarly, Gschwind and Irnich [18] incorporated synchronization constraints and adapted known cuts for their BPC framework. In the context of multi-trip DARPs, Luo et al. [19] proposed a two-phase algorithm. The first phase builds a trip-based model by enumerating a pool of non-dominated

trips using a label-setting algorithm; the second phase applies BPC combined with infeasible path cuts and Benders cuts to ensure global route feasibility. Recently, a BCP algorithm to solve an equitable variant of the DARP, namely equity-aware DARP, a bi-objective optimization problem that simultaneously minimizes the total routing cost and maximizes the equity-of-travel outcomes for individual passengers [66]. Despite the significant progress, it is noteworthy that most exact methods remain limited to small- and medium-scale instances. As of recent studies, the largest DARP instances solved to proven optimality include no more than 100 transportation requests [63].

To address larger-scale and more complex DARPs, researchers have turned to heuristic and metaheuristic methods, which provide high-quality solutions within reasonable computational times, especially when exact methods become intractable. Metaheuristics such as tabu search (TS), variable neighborhood search (VNS), large neighborhood search (LNS), genetic algorithms (GA), and hybrid frameworks have been widely used. Cordeau and Laporte [17] were among the first to apply TS to the DARP, introducing memory structures and diversification strategies to escape local optima. Their framework has inspired a generation of TS-based methods with modifications in neighborhood operators, penalty mechanisms, and search intensification procedures. The use of this algorithm in other studies (e.g., [27, 67, 68]) further supports its applicability and effectiveness.

VNS-based algorithms have been particularly influential due to their adaptability and performance across different DARP variants. These methods systematically explore an increasing set of neighborhood structures and have been applied to both single- and multi-objective problems [16, 29]. Recent studies have introduced evolutionary VNS, which employs a population-based approach and incorporates genetic operators to enhance solution diversity [42]. Similarly, LNS algorithms, which alternate between destruction and repair phases, have demonstrated success in balancing intensification and diversification [69, 70]. Adaptive versions of LNS have been integrated with learning mechanisms to dynamically adjust removal and insertion strategies [14, 35, 71]. GA has also proven effective in solving DARPs, especially when integrated with local search components. Examples include cluster-first, route-second methods that use GA for clustering and heuristics for routing [26, 72].

In addition to these methods, hybrid algorithms that use the capabilities of multiple algorithms have been very popular. This combination can be a hybrid of two or more (meta-) heuristics, a hybrid of metaheuristic and mathematical model approaches, and a hybrid of metaheuristic and constraint programming. One of the most common hybrid algorithms is a combination of population-based algorithms (e.g., GA) to better explore the solution space and trajectory-based algorithms (e.g., TS, VNS, and LNS) for proper exploitation. For ex-

ample, a hybrid GA algorithm with two crossover operators and four mutation operators was proposed for solving the heterogeneous DARP by Masmoudi et al. [73]. This hybrid algorithm applied local search with five well-known operators from the routing literature to the newly generated offspring in GA. Similarly, a simulated annealing method is presented as a trajectory-based algorithm embedded within a bee algorithm as a population-based one for solving the heterogeneous DARP with multiple depots [74]. Another way to create hybrids is to use a local search method in a metaheuristic algorithm, for example, the ruin-and-recreate method has been applied in the GRASP algorithm [24]. Belhaiza [75] combined an adaptive LNS algorithm with a GA structure to solve DARP with time windows. Souza et al. [55] proposed a two-phase heuristic algorithm for solving dynamic DARP. In the first phase, a general VNS algorithm and a randomized variable neighborhood descent method are combined to solve the static part of the problem, and a simple insertion heuristic is used for the dynamic part of the problem.

In summary, the literature on the DARP encompasses a wide range of exact and heuristic methods, each tailored to specific problem variants and scales. Exact algorithms have proven effective for small to medium-sized instances, offering optimal solutions under tightly defined constraints. However, numerous heuristic and metaheuristic algorithms have been developed to efficiently generate high-quality solutions within reasonable computation times.

2.2 Large-scale Routing Problem

The computational intractability of exact methods in solving large-scale routing problems has become increasingly apparent, especially in the context of the DARP, where the presence of time windows, precedence constraints, and multi-capacity vehicles results in a highly complex combinatorial structure [18]. This challenge has led to a growing interest in heuristic and metaheuristic approaches that offer more scalable and time-efficient solutions [11, 14, 33, 35, 40, 69, 76]. While these techniques have significantly improved the tractability of small- and medium-sized DARP instances, their extension to large-scale scenarios remains relatively underexplored. This is particularly critical given the real-world emergence of high-demand, on-demand transportation services in urban settings, where instances frequently involve thousands—or even hundreds of thousands—of user requests.

Early efforts to bridge this scalability gap include the work of Xiang et al. [77], who proposed a heuristic framework based on local search, enhanced by diversification and intensification strategies. Their solution approach, tested on instances with up to 2,000 requests, was among the first to demonstrate the feasibility of heuristic methods for managing the computational burden of large DARP instances while still achieving acceptable solution quality

within reasonable time limits. This contribution marked an important milestone in moving beyond exact optimization paradigms for realistic applications. Subsequent advances have focused on improving the scalability of metaheuristics through distributed computing and problem decomposition. Muelas et al. [78], building upon their earlier work [79], introduced a distributed VNS algorithm. Their approach partitions the request space into manageable subsets and then applies route combination techniques to coordinate the global solution. This methodology was evaluated on a large-scale case study based in San Francisco, involving up to 16,000 requests. Their results not only highlighted the computational efficiency of distributed search but also emphasized the value of spatial partitioning in mitigating the dimensionality curse often associated with real-world DARP instances.

More recently, Liu et al. [80] explored large-scale homogeneous DARP under a new operational paradigm: shared autonomous vehicle systems. They proposed a greedy insertion heuristic augmented with a filtering mechanism for decision acceleration and introduced a network-driven route encoding framework. The strength of their method lies in its capacity to handle extremely large instances, ranging from 10,000 to 300,000 requests across networks with up to 15,000 nodes. Their work showcases the integration of network analysis, algorithmic design, and modern transportation systems to address the rising demand for scalable on-demand mobility services.

Insights from the broader vehicle routing problem (VRP) literature have further enriched the toolkit available for large-scale DARP. VRP, as a more general class of routing problems, has seen extensive efforts in scaling algorithms to industrial-size instances. For example, Kytöjoki et al. [81] developed a VNS-based algorithm that combines several insertion heuristics to construct high-quality initial solutions, which are then refined using variable neighborhood descent. Their method solved instances with up to 20,000 customers in under an hour, demonstrating that careful hybridization of local search techniques can yield effective large-scale solvers. Similarly, Qi et al. [82] proposed a two-phase strategy for VRP with time windows. The cluster-first, route-second approach utilizes k-medoid clustering and GA to achieve spatial and temporal decomposition, enabling more tractable subproblem resolutions. These clustering strategies laid the groundwork for numerous later studies attempting to break down problem size before optimization. More recently, Arnold et al. [83] presented a scalable local search heuristic that can handle capacitated VRP instances with up to 30,000 customers. Their algorithm uses sequential search strategies and powerful pruning techniques to reduce the effective search space. This combination allows for efficient exploration while maintaining high solution quality, thereby offering practical relevance to large-scale logistics operations.

In the same manner, Accorsi and Vigo [84] developed the FILO algorithm, a hybrid iterated local search enriched with simulated annealing acceptance rules and acceleration techniques. FILO achieved high-quality results on benchmark instances of considerable size (up to 30,000 customers), with run times restricted to around 150 minutes. Their work illustrates the balance between solution quality and computational efficiency in solving massive-scale problems.

Recent studies have begun leveraging machine learning to enhance large-scale VRP methodologies. Zhang et al. [85] introduced a two-stage learning-based method comprising a clustering and routing phase. The clustering phase relies on graph convolutional networks and attention mechanisms to maintain pickup-delivery integrity while classifying requests for different vehicles. The routing phase then uses a trained deep learning model to generate feasible tours. Remarkably, their model, initially trained on small-scale instances, demonstrated strong generalization capabilities when applied to much larger instances, showcasing the potential of transfer learning in combinatorial optimization. Furthering the application of adaptive learning in routing, Máximo et al. [86] proposed AILS-II, an adaptive iterated local search framework. The algorithm operates in two phases, each incorporating perturbation and local search but differing in how reference solutions are selected—either via acceptance criteria or from an elite solution set. AILS-II outperformed state-of-the-art benchmarks on both small and large VRP instances, reaching up to 30,000 vertices and yielding superior solution quality and stability. Cavaliere et al. [87] address two practical variants of the VRP: the VRP with simultaneous pickup and delivery (VRPSPD) and the VRP with mixed pickup and delivery (VRPMPD). They introduced FSPD, a specialized extension of the FILO framework, designed to efficiently solve these variants. The algorithm demonstrates strong competitiveness with existing state-of-the-art methods while maintaining linear scalability in computational time, even on newly proposed large-scale benchmark instances with up to 30,000 customers. Accorsi and Vigo [88] extend their earlier work [84] by introducing a new dataset with significantly larger instance sizes (ranging from 20,000 to 1,000,000) to challenge current algorithmic capabilities. They present FILO2, an improved version of their FILO algorithm, which incorporates enhanced acceleration and pruning strategies. Tested on extremely large-scale instances, FILO2 delivers high-quality solutions efficiently, demonstrating superior scalability. While the instances exceed typical real-world applications, they serve as valuable benchmarks for advancing research on scalable routing algorithms.

Table 2.1 summarizes key studies on large-scale routing problems, outlining the problem type, algorithmic approach, largest problem size addressed, and corresponding computation time. The table highlights a trend toward increasing scalability through advanced heuristics, hybrid methods, and learning-based techniques, with different problem sizes and varying solution times depending on the algorithm used.

Table 2.1 Summary of large-scale routing problem studies

Reference	Problem Type	Algorithm	Largest Problem Size	Time (min)
Xiang et al. [77]	DARP	Local search strategies	2,000 requests	360
Muelas et al. [78]	DARP	Distributed VNS	16,000 requests	180
Kytöjoki et al. [81]	VRP	VNS	20,000 customers	144
Qi et al. [82]	VRP	Hybrid clustering and GA	1,000 customers	2
Arnold et al. [83]	VRP	Local search strategies	30,000 customers	600
Accorsi & Vigo [84]	VRP	FILO	30,000 customers	150
Zhang et al. [85]	VRP	Learning-based method	1,000 customers	15
Liu et al. [80]	DARP	Hybrid greedy insertion	300,000 requests	313
Máximo et al. [86]	VRP	Adaptive iterated local search	30,000 customers	1160
Cavaliere et al. [87]	VRPSPD / VRPMPD	FSPD	30,000 customers	28
Accorsi & Vigo [88]	VRP	FILO2	1,000,000 customers	167

Although significant progress has been made in the routing problem literature, relatively few studies have directly tackled the unique challenges associated with very large-scale and practical DARP instances. The complexity of such problems, characterized by high-dimensional search spaces and computationally expensive constraints, necessitates the development of advanced algorithms capable of balancing solution quality and computational efficiency.

2.3 Electric Dial-a-Ride Problem (E-DARP)

The growing adoption of EVs in transportation systems has prompted the need to revisit classical vehicle routing problems by incorporating energy-related constraints. One of the well-studied extensions in this context is the electric pickup and delivery problem with time windows (EPDPTW), which emerges when conventional vehicles are replaced with EVs in the traditional PDPTW. The EPDPTW introduces additional layers of complexity due to factors such as limited driving range, the need for recharging, and time-dependent energy consumption, all while maintaining service time constraints and precedence requirements between pickup and delivery locations.

Several studies have addressed the EPDPTW by proposing both exact and heuristic approaches tailored to the unique characteristics of electric fleets. Grandinetti et al. [89] formulated the EPDPTW as a multi-objective optimization problem, aiming to minimize the total travel distance, the cost associated with EV usage, and penalties incurred from service delays. Goeke [90] presented a compact model for the EPDPTW that incorporates a partial recharging policy and proposed a granular tabu search algorithm. Their approach effectively handles partial charging while allowing limited violations of time window constraints to enhance flexibility. In a stochastic setting, Soysal et al. [91] introduced a chance-constrained mixed-integer nonlinear programming model, considering the uncertainty in battery depletion. They also

developed a linear approximation method to ensure computational tractability while preserving solution quality. Liu et al. [92] studied the electric PDPTW under demand uncertainty using a two-stage adaptive robust optimization model. Routing and timing decisions are fixed in advance, while loading/unloading quantities adapt to actual demands. To solve the model efficiently, they developed a two-phase decomposition method combining a branch-and-price-and-cut algorithm and dynamic programming. Their results highlight the robustness of the proposed approach and the influence of battery consumption rates on routing performance. Agrali and Lee [93] introduced a variant of the EPDPTW that allows transfers between vehicles at recharging facilities, relaxing the traditional single-vehicle-per-request constraint. The model also incorporates multi-depots, time windows, and EV-specific constraints. A hybrid heuristic combining simulated annealing and LNS was proposed to solve the problem efficiently, achieving optimal solutions significantly faster than CPLEX on small instances. Recently, Zhou et al. [94] addressed the EPDPTW by incorporating queue scheduling for vehicles at shared locations. A MIP model and an adaptive hybrid neighborhood search algorithm were proposed. The approach proved effective, finding new best solutions and highlighting the importance of considering queue delays in EV logistics.

These advancements have significantly contributed to the broader understanding of how electrification affects routing and scheduling in logistics systems. However, the focus in EPDPTW remains largely on freight delivery, with limited attention to more passenger-centric applications. This has led to increased interest in the E-DARP, an extension of the DARP that accounts for electric vehicle constraints. The E-DARP incorporates the complexities associated with EV operations, including limited battery capacity, the need for recharging, and the impact of energy consumption on routing decisions. Over the last decade, a growing body of research has been dedicated to modeling and solving the E-DARP, proposing a range of exact and heuristic algorithms to address its unique challenges.

Masmoudi et al. [42] introduced the E-DARP in the context of non-emergency healthcare transportation. Their model integrates heterogeneous electric vehicles that provide various medical transport resources under capacity constraints. A key feature of their formulation is the use of battery swapping stations with fixed swap times, while travel-based energy consumption varies by arc. To solve the problem, they developed three versions of a hybrid evolutionary VNS algorithm embedded in a genetic framework. Their study stands out for adapting standard DARP benchmark datasets and generating a new set of artificial instances specifically designed for electric vehicle operations.

Building on the foundational structure of DARP, Bongiovanni et al. [15] proposed the electric autonomous DARP (E-ADARP), which considers electric autonomous vehicles operating

from multiple starting and ending depots. A novel constraint introduced in their model is the requirement for each vehicle to maintain a minimum SoC upon return, enabling more realistic modeling of battery usage. Moreover, they imposed access limitations on recharging stations—only empty vehicles may use them, and partial recharging is permitted. They developed both three-index and two-index mixed-integer programming (MIP) models and employed a BC algorithm with problem-specific valid inequalities. Their experimental validation included both benchmark-based and real-world ridesharing datasets, such as data derived from Uber Technologies Inc.

In a subsequent study, Bongiovanni et al. [95] addressed the dynamic nature of electric autonomous ridesharing and extended the static E-ADARP to its online variant. They proposed a two-phase machine learning-based LNS. The first phase comprises a greedy insertion heuristic for assigning real-time requests, and the second phase uses a metaheuristic based on destroy-repair strategies, where the choice of operators is guided by a machine learning model trained offline on over 1.5 million solved subinstances. This marks one of the first significant applications of supervised learning to guide neighborhood search in E-DARP.

Su et al. [96] introduced a novel fragment-based path representation for the E-ADARP, where path fragments are abstracted into arcs, enabling the construction of a sparse graph that retains all feasible original paths. This method allows optimality in minimizing excess user ride time while significantly improving computational efficiency. Their approach is implemented in a column generation framework, where a restricted master problem and pricing subproblems are solved iteratively. A tailored labeling algorithm, using resource extension functions and strong dominance rules, ensures the feasibility of partial recharging and guarantees optimal label extension for each path.

A complementary strategy was proposed by Limmer [97], who developed a bilevel LNS framework. In the outer level, charging sessions are inserted into vehicle routes, and in the inner level, pickup and delivery decisions are optimized. Their framework was tested on several instance sets, including extensions of the well-known benchmarks, and large-scale instances with up to 260 vehicles and 5200 requests. This study is notable for tackling large-scale instances while explicitly modeling the interaction between routing and charging strategies.

Addressing a different setting, Molenbruch et al. [43] studied the E-DARP on a fixed circuit, motivated by the operation of autonomous electric shuttles. This problem assumes a circular network where shuttles follow a fixed route and may stop flexibly for pickups and drop-offs. The objective is to minimize passenger excess time and total laps completed by the fleet, considering recharging needs between laps. They proposed a lap-based MIP formulation and developed specialized algorithms for subproblems, which were integrated into an LNS

metaheuristic. Their tests included both synthetic networks and a real-world case from Renmark, Australia.

Su et al. [98] advanced their previous work by introducing a deterministic annealing algorithm for E-ADARP. Their algorithm efficiently minimizes excess ride time using a linear-time route evaluation method and a fragment-based path representation. This study emphasizes exact evaluation under partial recharging and demonstrates the practical applicability of their method in dynamic scheduling environments. A notable recent contribution by Bresich et al. [99] developed an LNS heuristic that uses battery-restricted fragments for route representation and cost computation. Their method includes two charging-handling strategies: one that separates route, charging, and scheduling operations; and a second, integrated approach with dynamic charging stop insertion. Their approach achieved or improved best-known solutions for benchmark instances, demonstrating its high performance and flexibility.

Su et al. [100] proposed a BP algorithm leveraging a refined labeling approach for the path-based formulation of E-ADARP. Key innovations include a graph abstraction method that preserves feasible paths while reducing graph size, strong dominance rules, and feasibility checks with constant time complexity. Their method solved test instances to proven optimality, setting a new benchmark for exact solution methods in this field. A different modeling approach was introduced by Stallhofer & Parragh [101], who proposed an event-based MILP model for E-ADARP. To enhance the formulation, they introduced valid cuts and compared the performance of the new model against the BC algorithm of Bongiovanni et al. [15] and the column generation method of Su et al. [96]. Their findings highlight the potential of event-based modeling in capturing detailed temporal and energy-related constraints more precisely.

More recently, Dong et al. [1] explored the integration of E-DARP with time-of-use electricity pricing and ridesharing. Their model incorporates partial charging, time-dependent charging costs, and user ride time penalties. They formulated the problem as a MIP and proposed a customized adaptive LNS algorithm that combines a dynamic programming module for charging optimization and a fast feasibility-checking procedure. Their tests on adapted DARP benchmarks and real-world datasets from electric taxi operations in Shenzhen demonstrate the model's practical relevance and flexibility in modern smart grid environments.

Table 2.2 provides a summary of these studies on the E-DARP, capturing problem and algorithmic strategies. As observed, while early works focused on small to medium-sized instances, more recent efforts address larger networks. This table also highlights the diversity of solution methodologies and indicates a growing effort to tackle the computational challenges posed by large-scale, realistic E-DARP instances.

Table 2.2 Overview of recent contributions to the E-DARP

Reference	Problem	Type of algorithm	Largest instance size	
			Vehicles	Requests
Masmoudi et al. [42]	Static E-DARP	Heuristic	17	100
Bongiovanni et al. [15]	Static E-ADARP	Exact	5	50
Bongiovanni et al. [95]	Dynamic E-ADARP	Hybrid heuristic	5	60
Su et al. [96]	Static E-ADARP	Exact	5	50
Limmer [97]	Static E-ADARP	Heuristic	260	5,200
Molenbruch et al. [43]	Static E-DARP on a fixed circuit	Heuristic	8	300
Su et al. [98]	Static E-ADARP	Heuristic	8	96
Bresich et al. [99]	Static E-ADARP	Heuristic	8	96
Su et al. [100]	Static E-ADARP	Exact	8	96
Stallhofer & Parragh [101]	Static E-ADARP	Exact	8	96
Dong et al. [1]	Static E-DARP	Heuristic	900	10,000

Despite notable advancements in research on the E-DARP, many existing studies rely on simplifying assumptions that limit their real-world applicability. A common assumption is that all vehicles are equipped with homogeneous battery capacities and begin operations with fully charged batteries. Additionally, most models consider only a single type of charging infrastructure, often excluding the diversity of technologies such as fast or slow charging stations. This uniform treatment of infrastructure overlooks the operational flexibility and limitations associated with different charging technologies. Charging behavior is typically modeled using linear functions, which fail to capture the inherently nonlinear and state-dependent nature of battery charging processes. Moreover, operational constraints at charging stations—such as limited capacity, simultaneous charging restrictions, and potential queueing delays—are rarely incorporated into existing formulations. These constraints can significantly influence the feasibility and efficiency of routing plans, particularly in dense urban settings with limited infrastructure availability. The lack of attention to these realistic elements constrains the practical deployment of many proposed solutions.

CHAPTER 3 THESIS ORGANIZATION

As discussed in Chapter 2, a key gap in the existing literature lies in the limited availability of optimization methods that can handle (i) the modeling realism required by operational DARP settings, (ii) the computational demands of large and very large-scale instances, and (iii) the emerging challenges associated with the integration of EV fleets. Most prior studies either focus on simplified problem variants or employ algorithms that do not scale to real-world dimensions, especially for exact algorithms. Furthermore, features such as contractual cost structures, energy dynamics, and charging infrastructure constraints remain largely underexplored in large-scale DARP models.

To address this gap, this dissertation is organized into three core articles, each targeting a specific aspect of the problem. The order of these articles reflects a deliberate methodological progression: from exact approaches for medium-sized, high-fidelity problems to scalable heuristic methods for very large-scale and energy-constrained environments. This evolution mirrors the research trajectory followed during the doctoral project and highlights how each article builds upon the findings of the previous one.

The first article, presented in Chapter 4, focuses on the development of an exact BPC algorithm for solving DARP instances that reflect real operational conditions. The model includes multiple advanced features: heterogeneous users and vehicles, multi-resource capacity, contract-specific routing constraints and budget, time windows, maximum ride times, and break regulations. The proposed algorithm is capable of solving realistic small- to medium-scale instances (the largest DARP instance reported to be optimized in the literature) to proven optimality, as motivated by a real-world case study from GIRO Inc. Chapter 4 serves as the foundation of the dissertation by establishing a robust and flexible modeling framework and by demonstrating the value of exact methods in generating optimal solutions under realistic constraints.

Recognizing the computational limitations of exact methods when applied to large-scale instances, the second article (Chapter 5) shifts focus to heuristic approaches. It introduces a VNS algorithm tailored to address the complexity and size of real-world DARP datasets—comprising thousands of requests and hundreds of vehicles—well beyond the reach of exact methods. The algorithm incorporates a linear relaxation-based initialization method and a structured neighborhood design to efficiently explore large solution spaces. Chapter 5 builds directly upon the model and constraints introduced in Chapter 4, adapting them to a heuristic context and ensuring that high-quality solutions can be obtained within acceptable

computational times. In addition, the optimal solutions obtained with the exact algorithm of Chapter 4 allow us to assess the quality of the solutions produced by the heuristic of Chapter 5 on medium-sized instances.

The third article, shown in Chapter 6, extends the VNS framework to address the E-DARP, a novel and increasingly relevant variant of the classical DARP. Chapter 6 introduces energy constraints related to battery consumption, concave piecewise linear charging dynamics, time-of-use electricity pricing, and charging station capacities, aspects largely overlooked in prior studies. The heuristic is adapted to ensure feasibility under these new dimensions, while maintaining efficiency in large-scale contexts. Building on the scalable approach of Chapter 5, this contribution integrates environmental and infrastructure considerations into the planning model, making it applicable to future electric mobility systems.

The three articles are interlinked by a shared goal of improving decision-making in demand-responsive transportation systems under increasing operational complexity. Chapter 4 provides the modeling and optimization foundation; Chapter 5 introduces algorithmic scalability for large-scale deployment; and Chapter 6 adds the layer of electrification, responding to sustainability concerns and real-world charging constraints. This progression allows the dissertation to evolve from theoretical rigor to practical applicability, while maintaining methodological coherence across chapters.

In summary, this dissertation presents a unified and structured response to the limitations identified in the literature. It contributes novel models and solution methods that are both operationally grounded and computationally efficient, enabling the planning of next-generation dial-a-ride services that are responsive to user needs, scalable in size, and compatible with electric vehicle technologies.

CHAPTER 4 ARTICLE 1: AN EXACT BRANCH-AND-PRICE-AND-CUT ALGORITHM FOR A PRACTICAL AND LARGE-SCALE DIAL-A-RIDE PROBLEM

Authors: Mohammad Karimi, Fanny Camiat, Guy Desaulniers, Michel Gendreau

Note: Published in Journal of the Operational Research Society 76(6), 1125-1139, 2025 (Published online: 10 Oct 2024).

4.1 Introduction

In recent decades, the dial-a-ride problem (DARP) has gained significant attention, primarily driven by its application in real-world contexts like dial-a-ride, on-demand mobility, and paratransit services. The DARP builds upon the pickup and delivery problem (PDP, see [9]), which involves the transportation of freight with the same vehicle responsible for both pickup and drop-off and, more precisely, on the PDP with time windows (PDPTW, see [4, 102]), where each location has a particular interval of time for service initiation. While the PDPTW primarily concentrates on the transportation of goods, the DARP specifically examines passenger transportation and incorporates additional service-related requirements such as maximum ride-time constraints [8], in addition to common PDPTW constraints (e.g., pairing, precedence, capacity, and time window constraints). One of the main DARP applications arises from the patient transportation services provided by hospitals and other medical institutions [5–7], where the objective is to efficiently schedule a fleet of ambulances for the transportation of elderly individuals or patients between their homes and hospitals.

Various real-world features arising from different applications have been considered in the DARP. In this paper, we consider practical features that have not yet been studied in the literature and that arise from a real-world case proposed to us by our industrial partner GIRO Inc. Assume that several sets of vehicles are available according to different contracts, and define a *route* as a vehicle journey starting and ending at the depot and corresponding to the work done by a vehicle over a whole day. Each route may be subject to a maximum duration and a choice of break patterns dictated by the vehicle's contract. In addition to the limited number of vehicles available per contract, the budget allocated to each contract is limited, according to a specific cost structure. We model this practical version of the DARP as an integer linear program with an exponential number of variables and develop an exact branch-price-and-cut (BPC) algorithm to solve it. To tackle the column generation

subproblems, we devise an ad hoc labeling algorithm that can handle all feasibility rules of the vehicle routes. The proposed algorithm successfully optimizes a set of real-world instances, including the largest case with 849 requests and over 70 available vehicles managed under 5 different contracts. To our knowledge, this is the largest DARP instance ever solved to optimality in the literature. Solving this instance also allows to assess the performance of the heuristic commercialized by GIRO.

The remainder of this paper is organized as follows. We review the studies related to the DARP in Section 4.2. Section 4.3 provides a detailed description of the DARP of interest. In Section 4.4, we model this problem as an integer program with a large number of variables. Then, in Section 4.5, we describe the proposed BPC algorithm, including the labeling algorithm, valid inequalities, and branching strategies. Section 4.6 introduces the data adopted in the case study and presents our computational results. Finally, conclusions are drawn in Section 4.7.

4.2 Literature review

Before surveying the solution algorithms proposed to solve the DARP and its many variants, we briefly list problem features that have been tackled in the literature. Papers focussing on algorithmic developments like [11, 31–33] often consider a problem setting that includes a homogeneous fleet of capacitated vehicles, pickup and delivery time windows, maximum route length, and maximum passenger ride time. Several DARP applications and practical needs have motivated the consideration of other problem characteristics: heterogeneous passengers and vehicles [10,12,13,65], passenger transfer between vehicles [34–36,103], use of a fixed route and scheduled public transport service during the user's journey [37–39], need for additional manpower [13, 23, 40], driver scheduling rules [19, 41], and electric vehicles [15, 42, 43], among others.

There exist various algorithms for solving the DARP variants. Most of the exact ones rely on branch-and-bound (BB). To our knowledge, Cordeau [61] proposed the first branch-and-cut (BC) algorithm for the DARP and introduced several families of valid inequalities, derived from well-known ones for the traveling salesman problem and the vehicle routing problem. Ropke et al. [104] introduced three new classes of valid inequalities and adopted some previously identified cuts. Other valid inequalities were derived for specific problem characteristics, such as trip number, lunch breaks [41], symmetry breaking, driver stability [105], and autonomous electric vehicles [15]. Cortés et al. [34] developed a BC method to solve the DARP with possible passenger transfers between vehicles, where combinatorial Benders cuts are generated to guarantee feasibility of the routes and users' journeys. More recently,

BC algorithms based on new problem formulations that implicitly handle vehicle capacity as well as pickup-and-delivery pairing and precedence constraints were designed: Rist & Forbes [62] proposed a formulation based on flow variables associated with route fragments (i.e., sequences of pickups and deliveries starting and ending with an empty vehicle) whereas Gaul et al. [63] introduced two formulations defined on a network with vertices representing each an event (pickup or delivery) together with a list of onboard requests.

Branch-and-price (BP) algorithms rely on a decomposition of the problem into a master problem that coordinates the selection of the routes to cover all requests and a pricing subproblem that generates new vehicle routes to consider. The linear relaxations in the BB search tree are then solved by a column generation (CG) algorithm, which alternates between solving a restricted version of the master problem and the pricing subproblem. For the DARP, the subproblem is usually formulated as an elementary shortest path problem with resource constraints and solved by dynamic programming [21]. In [22], the authors developed three heuristics for solving the pricing subproblem that are invoked before calling an exact labeling algorithm if necessary. To take advantage of both BP and BC methods, Qu & Bard [65] proposed a BPC algorithm that applies subset-row inequalities. They showed the benefits of these cuts on reducing the number of BB nodes and on the overall solution time. Considering intra-route synchronization constraints in their problem, Gschwind & Irnich [18] developed a BPC algorithm that adopts several classes of known inequalities. In a multi-trip version of the DARP, Luo et al. [19] introduced a two-phase algorithm relying on a strong trip-based model. This model is built in the first phase by enumerating a set of non-dominated trips via a label-setting algorithm. Then, it model solved by a BPC algorithm that generates infeasible path cuts and Benders cuts to ensure route feasibility. From this review of the exact algorithms, we observe that the largest DARP instances reported to be solved to optimality have 100 requests [63].

In the existing literature, different heuristics have been conceived to solve various DARP variants, such as tabu search [17], large neighborhood search [35, 69, 76], evolutionary local search [33], and hybrid algorithms [11, 14, 40]. Interested readers may refer to [106] for a comprehensive review of DARP.

Heuristics are, typically, developed to solve large-sized instances in relatively fast computational time. On the other hand, exact algorithms with the ability to reach an optimal solution are mostly used to solve small-sized instances. This paper presents a BPC algorithm capable of solving large-sized DARP instances, ranging from 300 to 849 requests, in less than 8 hours. This DARP instances are real-world ones that include several features that are new or have not been addressed together.

4.3 Problem definition

The DARP variant that we consider in this paper has the following characteristics. The transportation requests are heterogeneous, i.e., a passenger might be ambulant or in a wheelchair. Both passenger types require a seat, while the latter also need a space to transport their wheelchair. The passenger type affects the request service time. Each request has a time window at the pickup and drop-off point and a maximum ride time. Vehicles are available according to different contracts. The number of vehicles in each contract and the total cost that can be spent for each contract (defined as the contract budget) is limited. Each contract provides a vehicle type with specific capacities and a cost structure. As mentioned above, a *route* is defined as a sequence of activities, including a pull-out from the depot, pickups, drop-offs, breaks, and a pull-in into the depot. A *block* is a part of a route between two consecutive locations where the vehicle is empty. Thus, a route can contain one or several blocks. For some contracts, routes must not exceed a maximum duration and breaks must be considered according to different break patterns. The break pattern to apply depends on the route duration and indicates the number of breaks to assign and the admissible duration between the breaks or the route start/end. Each contract has its own cost structure. Because there are different vehicle types, the travel times and distances may vary from one vehicle to another, impacting the cost structure.

Let us provide a formal description of this practical DARP. The problem is defined on a directed graph $G = (V, A)$, where V is the set of vertices and A the set of arcs. Vertex set V is partitioned into two copies of the depot (0 and $2n + 1$), the set of pickup vertices $P = \{1, \ldots, n\}$, and the set of delivery vertices $D = \{n+1, \ldots, 2n\}$. Each pickup vertex $i \in P$ defines a request to transport passengers (typically, one but sometimes several) from i to the delivery vertex $n + i$. These passengers may demand one of two different transportation modes (ambulant or wheelchair). When a passenger boards a vehicle, it consumes some resources (a seat and possibly a folded wheelchair space). Let $M = \{St, Wc\}$ denote the set of these two resources and let q_i^m be the amount of resources m associated with a vertex i ($q_i^m \geq 0$ if $i \in P$, $q_{n+i}^m = -q_i^m \leq 0$ if $n + i \in D$, and $q_i^m = 0$ otherwise). Without loss of generality, the set of requests is represented by the set of pickup vertices P. Each request $i \in P$ must be executed without exceeding a maximum ride time denoted L_i. Furthermore, there is a time window $[e_i, u_i]$ associated with each vertex $i \in V$ that restricts the service start time at this vertex (or corresponds to the planning horizon if $i = 0$ or $2n + 1$).

The use of vehicles is defined by a set of contracts K. A contract $k \in K$ stipulates a number of available identical vehicles n_k, a total budget F_k for operating these vehicles, a cost structure and some operational rules. In particular, contract k specifies a vehicle capacity Q_k^m for each

resource $m \in M$ and a service time s_i^k at each pickup and delivery vertex $i \in P \cup D$. The travel time between two vertices $i, j \in V$ is also contract-dependent (it depends on the vehicle speed) and denoted t_{ij}^k. Similarly, the arc travel cost depends on the contract but also on other conditions to be described below. We assume that the travel times and costs satisfy the triangle inequality.

Arc set A can be restricted to the arcs that may be part of a feasible route. In fact, to represent the possible routes for contract $k \in K$, we rather introduce a graph $G^k = (V^k, A^k)$. Set $V^k \subseteq V$ excludes all pickup and delivery vertices that cannot be serviced by a vehicle in contract k, for example, if it has no capacity for a wheelchair. Denoting by $P^k = P \cap V^k$ and $D^k = D \cap V^k$ the sets of pickup and delivery vertices serviceable by contract k, arc set A^k is given by

$$\begin{aligned} A^k \;=\; & \{(0,i) \mid i \in P^k\} \cup \{(n+i, 2n+1) \mid n+i \in D^k\} \cup \\ & \{(i,j) \mid i \in D^k, j \in P^k, e_i + s_i^k + t_{ij}^k \leq u_j, i \neq j-n\} \cup \\ & \{(i,j) \mid i,j \in P^k, e_i + s_i^k + t_{ij}^k \leq u_j, q_i^m + q_j^m \leq Q_k^m, \forall m \in M\} \cup \\ & \{(i,j) \mid i,j \in D^k, e_i + s_i^k + t_{ij}^k \leq u_j, -q_i^m - q_j^m \leq Q_k^m, \forall m \in M\} \cup \\ & \{(i,j) \mid i \in P^k, j \in D^k, e_i + s_i^k + t_{ij}^k \leq u_j, q_i^m - q_j^m \leq Q_k^m, i \neq j-n, \forall m \in M\} \cup \\ & \{(i,j) \mid i \in P^k, j \in D^k, e_i + s_i^k + t_{ij}^k \leq u_j, q_i^m \leq Q_k^m, i = j-n, \forall m \in M\}. \end{aligned}$$

The time condition in the last four subsets excludes all arcs that cannot be traversed without violating the time window at vertex i or j, whereas the load condition in the last three subsets ensure that vehicle capacity is not exceeded by the passengers picked up or delivered at vertices i and j. The arcs in the first and second subsets are called the *pull-out* and *pull-in arcs*, respectively.

A route for a contract $k \in K$ is represented by a path from 0 to $2n+1$ in G^k together with a schedule for each visited vertex. It is feasible if it respects the pairing, precedence and maximum ride time constraints for each serviced request (i.e., if a pickup vertex $i \in P$ is visited, then its corresponding delivery vertex $n+i$ must also be visited afterwards but no later than L_i minutes after the service start time at i), the time window at each visited vertex, vehicle capacity for each available resource, and, possibly, a maximum route duration and break requirements. The break requirements are expressed through different break patterns that depend on the route duration. A break pattern ℓ is associated with a maximum route duration $T^{max,\ell}$ and indicates a number of breaks n^ℓ to assign (up to 3 in our case) and the duration d_b^ℓ of each break $b = 1, \ldots, n^\ell$. These breaks thus divide the route in $n^\ell + 1$ segments and the duration of segment $g = 1, \ldots, n^\ell + 1$ must fall in the interval $[\Delta_g^{min,\ell}, \Delta_g^{max,\ell}]$. It

should be mentioned, a break can only take place when there are no onboard passengers. We assume that this break takes place after unloading the last passenger.

As it is the case in practice, we also assume that the maximum time elapsed between two locations is always less than the minimum of the $\Delta_g^{min,\ell}$ values, implying that at most one break can be scheduled along an arc and none on the pull-out or pull-in arc of a route.

The cost of a route c_r^k depends on the contract $k \in K$ and is governed by two binary parameters, namely, w_k^{Pull} and w_k^{Dh} that specify whether or not the pull-ins/outs incur costs and the deadheads between the blocks incur costs, respectively. It is defined by:

$$\begin{aligned} c_r^k &= \gamma^{Blk} m_r^{Blk} + \gamma^{Dur}(\eta_r^{Pax} + \eta_r^{Brk} + w_k^{Pull}\eta_r^{Pull} + w_k^{Dh}\eta_r^{Dh}) \\ &\quad + \gamma^{Dst}(\chi_r^{Pax} + w_k^{Pull}\chi_r^{Pull} + w_k^{Dh}\chi_r^{Dh}), \end{aligned} \tag{4.1}$$

where m_r^{Blk} is the number of blocks in r; η_r^g is the total travel duration in r with at least one passenger onboard if $g = Pax$, of the mandatory breaks if $g = Brk$, of the pull-ins/outs if $g = Pull$, and of the deadheads between the blocks if $g = Dh$; similarly, χ_r^g is the total distance counterpart for $g = Pax, Pull, Dh$; and γ^g, $g = Blk, Dur, Dst$, are the unit costs per block, time, and distance, respectively. Penalizing the number of blocks helps avoid periods for which the vehicle is empty, thus, increasing the productive time of the vehicles. With this term and by managing travel costs and time effectively, more passengers can be accommodated within each block and avoid only direct trips between the pick-up and drop-off points of each request while adhering to time window and maximum ride time limits for each request.

Given this route cost that depends on the contract k and other conditions, we define the following cost c_{ij}^k for an arc $(i,j) \in A^k$:

$$c_{ij}^k = \begin{cases} \gamma^{Blk} + w_k^{Pull}(\gamma^{Dur}\eta_{ij}^{Pull} + \gamma^{Dst}\chi_{ij}^{Pull}) & \text{if } i = 0 \\ w_k^{Pull}(\gamma^{Dur}\eta_{ij}^{Pull} + \gamma^{Dst}\chi_{ij}^{Pull}) & \text{if } j = 2n+1 \\ \gamma^{Blk} + \gamma^{Dur}\eta_{ij}^{Brk} + w_k^{Dh}(\gamma^{Dur}\eta_{ij}^{Dh} + \gamma^{Dst}\chi_{ij}^{Dh}) & \text{if } (i,j) \text{ is a deadhead} \\ & \quad \text{with a break} \\ \gamma^{Blk} + w_k^{Dh}(\gamma^{Dur}\eta_{ij}^{Dh} + \gamma^{Dst}\chi_{ij}^{Dh}) & \text{if } (i,j) \text{ is a deadhead} \\ & \quad \text{without a break} \\ \gamma^{Dur}\eta_{ij}^{Pax} + \gamma^{Dst}\chi_{ij}^{Pax} & \text{if } (i,j) \text{ is traversed with} \\ & \quad \text{onboard passengers,} \end{cases} \tag{4.2}$$

where the arc parameters η_{ij}^g and χ_{ij}^g, $g = Pax, Pull, Dh$, are the counterparts of the route

parameters η_r^g and χ_r^g. The last three cases cannot be determined a priori, but rather dynamically when building routes in the labeling algorithm of Section 4.5.1.

4.4 Mathematical model

The DARP defined in the previous section can be formulated as an integer program with a very large number of variables using the following additional notation. Let R_k denote the set of feasible routes for the vehicles available according to contract $k \in K$. Although there may be multiple feasible schedules for each route, we assume that it is always associated with a least cost one. For each route $r \in R_k$, let c_r^k be its cost as defined by (5.1) and, for each request $i \in P$, let a_{ri} be a binary parameter indicating whether or not request i is serviced by route r. Finally, let y_r^k be a binary variable equal to 1 if and only if route $r \in R_k$, $k \in K$, is used in the solution.

The proposed integer programming formulation for the DARP is as follows:

$$\min \quad \sum_{k \in K} \sum_{r \in R_k} c_r^k y_r^k \tag{4.3}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{r \in R_k} a_{ri} y_r^k = 1, \qquad \forall i \in P \tag{4.4}$$

$$\sum_{r \in R_k} y_r^k \leq n_k, \qquad \forall k \in K \tag{4.5}$$

$$\sum_{r \in R_k} c_r^k y_r^k \leq F_k, \qquad \forall k \in K \tag{4.6}$$

$$y_r^k \in \{0, 1\}, \qquad \forall k \in K, r \in R_k. \tag{4.7}$$

Objective function (4.3) minimizes the total routing costs. Constraints (4.4) ensure that each request is covered by exactly one route. Constraints (4.5) and (4.6) impose vehicle availability and budget restriction for each contract, respectively. Finally, constraints (4.7) restrict the domain of the decision variables. Because this model contains a huge number of variables in practice (one per feasible route), we devise a BPC algorithm for solving it, which is presented next.

4.5 The proposed BPC algorithm

A BPC algorithm [107, 108] is a branch-and-bound algorithm where the linear relaxations are solved by column generation (CG) and cutting planes are added to strengthen the linear relaxations. CG is an iterative process that solves the linear relaxation of (4.3)–(4.7), which is called the *master problem* and is altered in the search tree by the applicable cutting planes

and branching decisions. At each CG iteration, the master problem restricted to a subset of its columns (variables) is first solved to compute a pair of optimal primal and dual solutions. To determine if this primal solution can be extended to an optimal solution for the whole master problem by setting the unknown variables to zero, one or several *pricing problems* are solved. In our case, there is one pricing problem per contract, namely, an elementary shortest path problem with resource constraints (ESPPRC) that is solved by a labeling algorithm. The goal of this pricing problem is to identify new columns with a negative reduced cost with respect to the current dual solution. If no such variables can be found for any of the pricing problems, then the CG process stops with an optimal primal solution. Otherwise, the negative reduced cost columns found are added to the restricted master problem to start a new iteration.

Below, we describe the labeling algorithm used to solve the pricing problems (Section 4.5.1), before presenting the cuts considered (Section 4.5.2) and our branching strategy (Section 4.5.3).

4.5.1 Labeling algorithm

As mentioned above, the pricing problem for contract $k \in K$ aims at finding negative reduced cost columns (feasible vehicle routes) for contract k. The reduced cost $\bar{c}_r^k$ of a column associated with a route $r \in R_k$ is given by:

$$\bar{c}_r^k = c_r^k(1-\beta^k) - \sum_{i \in P} a_{ri}\pi_i - \sigma^k = \sum_{(i,j)\in r} \bar{c}_{ij}^k, \tag{4.8}$$

where π_i, $i \in P$, σ^k, and β^k denote the dual variables associated with constraints (4.4)–(4.6), respectively, and the adjusted arc costs $\bar{c}_{ij}^k$ are given by:

$$\bar{c}_{ij}^k = \begin{cases} c_{ij}^k(1-\beta^k) - \pi_j - \sigma^k & \text{if } i = 0 \text{ and } j \in P^k \\ c_{ij}^k(1-\beta^k) - \pi_j & \text{if } i \neq 0 \text{ and } j \in P^k \\ c_{ij}^k(1-\beta^k) & \text{otherwise.} \end{cases} \tag{4.9}$$

Recall that the arc costs c_{ij}^k are not all known a priori (see Section 4.3) and, thus, the adjusted arc costs $\bar{c}_{ij}^k$ also.

The pricing problem is formulated as an ESPPRC on the graph G^k using the adjusted arc costs and enforcing route feasibility through resource constraints [108, 109]. To solve it, we propose a labeling algorithm. In such an algorithm, a partial path originating from the source vertex 0 is represented by a vector of attributes called a *label*, where each component indicates the value of a resource (including the reduced cost) at the end of the partial path. Starting

from an initial label at vertex 0, the algorithm generates partial paths by extending this label forwardly in graph G^k, checking the resource constraints after each extension to guarantee path feasibility. To avoid enumerating all feasible paths, a dominance rule is used to compare the partial paths ending at the same vertex. The negative reduced cost paths reaching the sink vertex $2n + 1$ become candidates to be added to the restricted master problem.

We describe the proposed labeling algorithm in steps, starting with a basic version that does not consider the maximum ride time constraints, the break requirements, and the maximum route duration. The handling of these features is discussed in Subsections 4.5.1, 4.5.1, and 4.5.1, respectively.

In the basic version, a label $l_p = (l_p^{rCost}, [l_p^{load(m)}]_{m \in M}, l_p^{time}, l_p^{open}, l_p^{unRch})$ representing a partial path p ending at vertex $i \in V^k$ is defined by the following attributes:

l_p^{rCost}**:** Reduced cost;

$l_p^{load(m)}$**:** Load for resource $m \in M$ at vertex i;

l_p^{time}**:** Earliest start of service time at vertex i;

l_p^{open}**:** Subset of open requests at vertex i, i.e., requests for which their pickup vertex has been visited but not their delivery vertex;

l_p^{unRch}**:** Subset of vertices that cannot be reached anymore because they have already been visited or because their time window cannot be met.

The labeling algorithm starts with an initial label representing the partial path p_0 containing only the source vertex 0: $l_{p_0} = (0, [0]_{m \in M}, e_0, \emptyset, \emptyset)$. This label is then recursively extended through graph G^k. The extension of a label l_p representing a path ending at vertex $i \in V^k$ through an arc $(i, j) \in A^k$ is considered only if $j \notin l_p^{unRch}$ and ($j \neq 2n + 1$ or $l_p^{open} = \emptyset$). It yields a new label $l_{p'}$ representing the path obtained by appending (i, j) to p. This label is computed using the following resource extension functions.

$$
\begin{aligned}
l_{p'}^{rCost} &= l_p^{rCost} + \bar{c}_{ij}^k && (4.10)\\
l_{p'}^{time} &= \max\{l_p^{time} + s_i^k + t_{ij}^k, e_j\} && (4.11)\\
l_{p'}^{load(m)} &= l_p^{load(m)} + q_j^m, \qquad \forall m \in M && (4.12)\\
l_{p'}^{open} &= \begin{cases} l_p^{open} \cup \{j\} & \text{if } j \in P^k \\ l_p^{open} \setminus \{j-n\} & \text{if } j \in D^k \\ l_p^{open} & \text{otherwise} \end{cases} && (4.13)\\
l_{p'}^{unRch} &= l_i^{unRch} \cup \{j\} \cup \{h \in V^k \mid l_{p'}^{time} + s_j^k + t_{j,h}^k > u_h\}. && (4.14)
\end{aligned}
$$

In (4.10), the adjusted arc cost $\bar{c}_{ij}^k$ is defined by (4.9) where the arc cost c_{ij}^k is determined according to (5.2). In this formula, arc (i, j) is identified as a deadhead if and only if $i, j \in P^k \cup D^k$ and $l_p^{open} = \emptyset$. Label $l_{p'}$ resulting from this extension is deemed feasible if $l_{p'}^{load(m)} \leq Q_k^m$ for all $m \in M$ and $o + n \notin l_{p'}^{unRch}$ for all $o \in l_{p'}^{open}$. Otherwise, $l_{p'}$ is infeasible and rejected.

The dominance rule applied for the complete problem version is presented in Subsection 4.5.1 after presenting the handling of the additional problem features.

Maximum passenger ride time

To consider the maximum ride time constraints, we should add to a label l_p a new resource (a function) for each open request $o \in l_p^{open}$:

$l_p^{lDT(o)}(t)$**:** latest possible delivery time of request o as a function of the start of service time t at the end vertex of p.

Gschwind & Irnich [18] proved that, instead of computing the entire function $l_p^{lDT(o)}(t)$, it is sufficient to keep track of its value at only two typically distinct times, namely, at l_p^{time} and at the time when $l_p^{lDT(o)}(t)$ becomes constant that we denote $l_p^{cstT(o)}$. Thus, following their proposal, only the following three new pieces of information for each open request o must be stored in a label l_p: $l_p^{cstT(o)}$, $l_p^{lDTt(o)} = l_p^{lDT(o)}(l_p^{time})$, and $l_p^{lDTc(o)} = l_p^{lDT(o)}(l_p^{cstT(o)})$. When extending a path p with an arc $(i, j) \in A^k$ to yield a path p', these label components are computed as follows:

$$l_{p'}^{cstT(o)} = \begin{cases} \min\{\tau_p^j, u_{j+n} - s_j^k - L_j\} & \text{if } o = j, \\ \max\{l_{p'}^{time}, \min\{\tau_p^j, l_p^{cstT(o)} + s_i^k + t_{ij}^k\}\} & \text{otherwise} \end{cases} \quad \forall o \in l_{p'}^{open} \tag{4.15}$$

$$l_{p'}^{lDTt(o)} = \begin{cases} \min\{l_{p'}^{time} + s_j^k + L_j, u_{j+n}\} & \text{if } o = j, \\ l_p^{lDTt(o)} + \min\{l_{p'}^{time} - s_i^k - t_{ij}^k, l_p^{cstT(o)}\} - l_p^{time} & \text{otherwise} \end{cases} \quad \forall o \in l_{p'}^{open} \tag{4.16}$$

$$l_{p'}^{lDTc(o)} = \begin{cases} l_{p'}^{cstT(o)} + s_j^k + L_j & \text{if } o = j, \\ l_p^{lDTc(o)} - \max\{0, l_p^{cstT(o)} + s_i^k + t_{ij}^k - \tau_p^j\} & \text{otherwise} \end{cases} \quad \forall o \in l_{p'}^{open} \tag{4.17}$$

where

$$\tau_p^j = \begin{cases} u_j & \text{if } j \in P \\ \min\{u_j, l_p^{lDTc(j-n)}\} & \text{if } j \in D \end{cases} \tag{4.18}$$

is the latest feasible start of service time at vertex j when reached from path p. This extension is feasible only if $l_{p'}^{time} \leq l_p^{lDTc(o)}$ for all $o \in l_p^{open}$. For further details, see [18].

Break requirements

Recall from the problem definition in Section 4.3 that breaks must be inserted in the routes of the vehicles governed by certain contracts. Those breaks must respect one of the admissible break patterns for this contract. Such a pattern ℓ specifies the number n^ℓ of breaks to insert, the duration d_b^ℓ of each break $b = 1, \ldots, n^\ell$, and the minimum and maximum durations $\Delta_g^{min,\ell}$ and $\Delta_g^{max,\ell}$ of the segments $g = 1, \ldots, n^\ell + 1$ induced by the breaks. To handle the break patterns, we replicate the subproblem associated with a contract subject to break requirements for each admissible break pattern. Next, we describe how the labeling algorithm is adapted to generate routes respecting a given break pattern. Note that, in this section, we assume that the start time of a route is fixed to the latest time that a vehicle can leave the depot vertex 0 to arrive exactly at the time window lower bound e_j of its first visited vertex $j \in P$. In the next section, we will discuss how to relax this assumption.

Let us consider the subproblem associated with a contract $k \in K$ and a break pattern ℓ that requires n^ℓ breaks. To enforce this break pattern, we extend the definition of a label l_p, associated with a partial path p ending at vertex $i \in V^k$, to include the following two resources:

l_p^{sgm}**:** the segment number at vertex i (an integer in $\{1, \ldots, n^\ell + 1\}$);

l_p^{durS}**:** the duration of the current segment l_p^{sgm}.

In the initial label l_{p_0} at vertex 0, these two components are set to $l_{p_0}^{sgm} = 1$ and $l_{p_0}^{durS} = 0$. Then, we can consider two possible extensions of a label l_p along an arc $(i, j) \in A^k$. The first extension does not include a break along this arc and yields a label $l_{p'}$ whose components are computed using the extension functions (4.10)–(4.17) and the following ones:

$$l_{p'}^{sgm} = l_p^{sgm} \tag{4.19}$$

$$l_{p'}^{durS} = \begin{cases} t_{ij}^k & \text{if } i = 0 \\ l_p^{durS} + l_{p'}^{time} - l_p^{time} & \text{otherwise.} \end{cases} \tag{4.20}$$

Beside the conditions for discarding a label previously presented, label $l_{p'}$ is also discarded if $l_{p'}^{durS} > \Delta_{l_{p'}^{sgm}}^{max,\ell}$ or ($j = 2n + 1$ and ($l_{p'}^{sgm} \neq n^\ell + 1$ or $l_{p'}^{durS} < \Delta_{l_{p'}^{sgm}}^{min,\ell}$)).

The second extension of label l_p along an arc $(i, j) \in A^k$ includes a break along this arc, namely, the break l_p^{sgm} of pattern ℓ. Assuming that the break would start at time $l_p^{time} + s_i^k$, this extension is considered only if $i \neq 0$, $j \neq 2n + 1$, $l_p^{sgm} \leq n^\ell$, $l_p^{open} = \emptyset$ and $\Delta_{l_p^{sgm}}^{min,\ell} \leq l_p^{durS} + s_i^k \leq \Delta_{l_p^{sgm}}^{max,\ell}$. It yields a label $l_{p''}$ that is computed using the extension functions (4.10), (4.12)–(4.17), and the following ones:

$$l_{p''}^{sgm} = l_p^{sgm} + 1 \tag{4.21}$$

$$l_{p''}^{time} = \max\{l_p^{time} + s_i^k + t_{ij}^k + d_{l_p^{sgm}}^\ell, e_j\} \tag{4.22}$$

$$l_{p''}^{durS} = l_{p''}^{time} - (l_p^{time} + s_i^k + d_{l_p^{sgm}}^\ell). \tag{4.23}$$

Notice that, for this case, the extension of the reduced cost component in (4.10) uses, in the adjusted reduced cost $\bar{c}_{ij}^k$, the arc cost c_{ij}^k for an arc containing a break as defined in (5.2). Because $l_p^{open} = \emptyset$ and we have assumed that a break cannot take place on a pull-in/out arc of a route (see Section 4.3), label $l_{p''}$ is discarded only if $l_{p''}^{time} > u_j$.

Maximum route duration

As suggested in [13, 22], to impose the maximum route duration $T^{max,\ell}$ associated with a given break pattern ℓ, we add the following two resources to every label l_p (representing a path p):

l_p^{wait}**:** the cumulated waiting time before the opening of a time window along path p;

l_p^{fwSl}**:** the forward slack time, i.e., by how much time the start at vertex 0 can be delayed while maintaining time feasibility along path p.

In the initial label p_0 at vertex 0, these components are set to $l_{p_0}^{wait} = 0$ and $l_{p_0}^{fwSl} = u_0 - e_0$. Let us consider the two possible extensions (without and with a break) of a label l_p along an arc $(i, j) \in A^k$ for a contract $k \in K$ and a break pattern ℓ with n^ℓ breaks (only the first extension is applicable for a contract without break requirements). The extension without a break yields a label $l_{p'}$ whose components are computed using the extension functions (4.10)–(4.17), (4.19)–(4.20) and

$$l_{p'}^{wait} = l_p^{wait} + \max\{0, e_j - (l_p^{time} + s_i^k + t_{ij}^k)\} \tag{4.24}$$

$$l_{p'}^{fwSl} = \min\{l_p^{fwSl}, l_p^{wait} + u_j - (l_p^{time} + s_i^k + t_{ij}^k)\}. \tag{4.25}$$

The extension with a break produces a label $l_{p''}$ which is computed using the extension functions (4.10), (4.12)–(4.17), (4.21)–(4.23) and

$$l_{p''}^{wait} = \omega_p^{ij} + \max\{0, e_j - (l_p^{time} + s_i^k + t_{ij}^k + d_{l_p^{sgm}}^\ell)\} \tag{4.26}$$

$$l_{p''}^{fwSl} = \min\{l_p^{fwSl}, \omega_p^{ij} + u_j - (l_p^{time} + s_i^k + t_{ij}^k + d_{l_p^{sgm}}^\ell)\}, \tag{4.27}$$

where

$$\omega_p^{ij} = \min\{l_p^{wait}, l_p^{durS} + s_i^k - \Delta_{l_p^{sgm}}^{min,\ell}\} \tag{4.28}$$

specifies the maximum cumulated waiting time before the break that can be used to decrease route duration (by delaying the route start time) without violating the minimum segment duration before the break.

Each resulting label l_ρ, $\rho = p', p''$, is discarded because of the maximum route duration constraint if $l_\rho^{time} + s_j^k + t_{j,2n+1}^k + d(l_\rho^{sgm}) - e_0 - l_\rho^{fwSl} > T^{max,\ell}$, where the sum of the remaining break durations is $d(l_\rho^{sgm}) = \sum_{b=l_\rho^{sgm}}^{n^\ell} d_b^\ell$ if $l_\rho^{sgm} \leq n^\ell$ and 0 otherwise.

Dominance rule

For the whole problem, we apply the following dominance rule that compares two labels l_{p_1} and l_{p_2} associated with paths ending at the same vertex. Label l_{p_1} dominates label l_{p_2} if the

following conditions hold:

$$l_{p_1}^{rCost} \leq l_{p_2}^{rCost} \tag{4.29}$$

$$l_{p_1}^{time} \leq l_{p_2}^{time} \tag{4.30}$$

$$l_{p_1}^{open} = l_{p_2}^{open} \tag{4.31}$$

$$l_{p_1}^{unRch} \subseteq l_{p_2}^{unRch} \tag{4.32}$$

$$l_{p_1}^{lDTt(o)} + (l_{p_2}^{time} - l_{p_1}^{time}) \geq l_{p_2}^{lDTt(o)}, \qquad \forall o \in l_{p_1}^{open} \tag{4.33}$$

$$l_{p_1}^{lDTc(o)} \geq l_{p_2}^{lDTc(o)}, \qquad \forall o \in l_{p_1}^{open} \tag{4.34}$$

$$l_{p_1}^{sgm} = l_{p_2}^{sgm} \tag{4.35}$$

$$l_{p_1}^{durS} \leq l_{p_2}^{durS} \tag{4.36}$$

$$\min\{l_{p_1}^{durS}, \Delta_{l_{p_1}^{sgm}}^{min,\ell}\} \geq \min\{l_{p_2}^{durS}, \Delta_{l_{p_2}^{sgm}}^{min,\ell}\} \tag{4.37}$$

$$l_{p_1}^{wait} \geq l_{p_2}^{wait} \tag{4.38}$$

$$l_{p_1}^{fwSl} \geq l_{p_2}^{fwSl}. \tag{4.39}$$

The above conditions ensure that, for every feasible extension of label l_{p_2}, there exists at least one feasible extension for label l_{p_1} that yields a better or equal reduced cost. Most of them were proposed in previous works (see references above). We only discuss the break-related conditions (4.35)–(4.37), and the open request condition (4.31). The break-related conditions as well as the maximum route duration conditions (4.38)-(4.39) are necessary only when the contract associated with the subproblem imposes break requirements. Condition (4.35) restricts dominance between labels that have the same number of breaks already scheduled. Condition (4.36), together with condition (4.38), ensures that label l_{p_1} will meet the maximum duration of the current segment if it can be met by label l_{p_2}. Furthermore, condition (4.37) guarantees that label l_{p_1} is in a better position than label l_{p_2} to respect the minimum duration of the current segment. As in the early BPC algorithms for pickup-and-delivery problems, we use an equality in condition (4.31) as opposed to the relaxed condition (4.40) discussed below. This very restrictive condition is necessary because of the segment minimum duration constraint. Indeed, it might become advantageous to perform additional deliveries to respect this constraint.

On the other hand, when no break requirements are enforced in a subproblem, condition (4.31) is replaced by

$$l_{p_1}^{open} \subseteq l_{p_2}^{open} \tag{4.40}$$

to improve dominance under the assumption that the so-called delivery triangle inequality holds for the adjusted arc costs. This is accomplished by transferring for every delivery vertex $n + i \in D^k$ a sufficiently large constant from the adjusted reduced cost of all arcs leaving $n+i$ to all arcs leaving pickup vertex i (for details [4]). Given that not all adjusted arc costs $\bar{c}^k_{ij}$ are determined a priori, we use worst-case costs to compute the constant to use for each delivery vertex.

Any dominated label can be discarded, except when multiple labels dominate each other. In this case, one of them must be kept.

Heuristic labeling

Given that the ESPPRC is $\mathcal{NP}$-hard in the strong sense [110], it is well-known that the exact labeling algorithm described above can be very time-consuming for large-sized networks. To alleviate this, heuristics can be used to generate columns before resorting to an exact labeling algorithm if needed. As proposed by several other authors [108], we apply two heuristic versions of the proposed labeling algorithm that eliminate a larger number of labels through a relaxed dominance rule. In the first heuristic, to determine if a label $l_{p'}$ can be discarded based on a comparison with a label l_p, we only consider the conditions (4.29)–(4.30) and (4.40) on the reduced cost, time, and open request resources. In the second, we use the conditions (4.29)–(4.39), except that we replace condition (4.31) by (4.40), including the break-related conditions only for the appropriate subproblems. The rest of both heuristic labeling algorithms, namely, label extensions and feasibility checks, remains unchanged.

Considering that there are several subproblems to solve at each CG iteration, the first labeling heuristic is first used for each subproblem. When it fails to find negative reduced cost columns for all subproblems, the second labeling heuristic is called for each subproblem. Finally, if this algorithm also fails to find promising columns, the exact labeling algorithm is invoked to ensure that the subproblems are solved to optimality. Note that the second labeling algorithm is, in fact, exact for the subproblems without break requirements and, therefore, there is no need to solve them again a third time.

4.5.2 Valid inequalities

To strengthen the linear relaxations encountered in the search tree, violated valid inequalities are added. We apply two families of valid inequalities, namely, rounded capacity inequalities (RCIs) and subset row inequalities (SRIs), as described next. The RCIs, which have been used for several vehicle routing problems [108] including the PDPTW [104], are defined as

follows. Let $U \subset P \cup D$ be a subset of vertices and $\xi_m(U)$ a lower bound on the number of vehicles needed to service all vertices in U with a demand for resource $m \in M$. The RCI associated with U and resource m is:

$$\sum_{k \in K} \sum_{r \in R_k} \sum_{(i,j) \in \delta^+(U)} a_{rij} y_r^k \geq \xi_m(U) \tag{4.41}$$

where a_{rij} is a binary parameter equal to 1 if route $r \in R^k$ traverses arc $(i,j) \in A^k$ and 0 otherwise; $\delta^+(U)$ is the subset of arcs in A^k that enters subset U;
$\xi_m(U) = \max\left\{1, \left\lceil \frac{\sum_{i \in P(U)} q_i^m}{Q^m} \right\rceil, \left\lceil \frac{-\sum_{i \in D(U)} q_i^m}{Q^m} \right\rceil \right\}$; Q^m is the maximum vehicle capacity for resource m over all contracts; $P(U) = \{i \in P \mid i \notin U, n+i \in U\}$ denotes the set of predecessors of U and $D(U) = \{n+i \in D \mid i \in U, n+i \notin U\}$ the set of successors of U. The numerator of the second fraction of $\xi_m(U)$ is a lower bound on the load of the vehicles entering U, and that of its last fraction is a lower bound on the load of the vehicles leaving U. For the separation of the RCIs, the heuristic separation procedure of [4] is used. It starts from a set $U = \{i\}$ with a single vertex i that is enlarged iteratively by adding one pickup or delivery vertex at a time. This vertex is chosen so as to maximize a parameterized objective function that seeks to find a violated RCI. This greedy procedure is repeated several times for each resource, possible initial vertex i, and randomly chosen objective function weights. All violated RCIs are added to the current linear relaxation and yield new dual variables that are handled in the adjusted arc costs of the subproblems e.g., [108].

When no violated RCIs can be found, we search for violated SRIs that are defined as follows. For a subset of requests $U \subseteq P$ and an integer λ ($1 < \lambda \leq |U|$), the SRI with respect to U and λ is:

$$\sum_{k \in K} \sum_{r \in R_k} \left\lfloor \frac{1}{\lambda} \sum_{i \in U} a_{ri} \right\rfloor y_r^k \leq \left\lfloor \frac{|U|}{\lambda} \right\rfloor. \tag{4.42}$$

Introduced first for the vehicle routing problem with time windows by Jepsen et al. [111], the SRIs are rank-1 Chvátal-Gomory inequalities valid for the general set-partitioning polytope. As suggested by several authors, we consider only the SRIs defined for $\lambda = 2$ and $|U| = 3$, which impose that at most one route covering more than one request in U be selected. The separation of the SRIs is done by pure enumeration and all cuts found are added to the current master problem. Handling their dual variables is non trivial and requires additional label components (for details, see [108, 111]).

4.5.3 Branching strategy

The proposed BPC algorithm adopts two types of branching decisions in a hierarchical structure. Both decision types are applied without changes to the problem structure: Linear constraints are added to the master problem accordingly and the associated dual variables are incorporated in the adjusted arc reduced costs.

Let $\left(\hat{y}_r^k\right)_{k\in K, r\in R_k}$ denote the master problem solution obtained at a node of the search tree that cannot be pruned. Then, if the total number of vehicles used $\hat{\nu} = \sum_{k\in K}\sum_{r\in R_k}\hat{y}_r^k$ is fractional, we branch on this entity and impose $\sum_{k\in K}\sum_{r\in R_k} y_r^k \leq \lfloor\hat{\nu}\rfloor$ on one branch and $\sum_{k\in K}\sum_{r\in R_k} y_r^k \geq \lceil\hat{\nu}\rceil$ on the other.

Otherwise, if the total number of vehicles used is integer, we rather branch, as proposed in [18], on the outflow of a vertex set $U \subset P \cup D$ such that $|U| = 2$. The outflow of U can be expressed as $\sum_{k\in K}\sum_{r\in R_k}\sum_{i\in U}\sum_{j\in V\setminus U} a_{rij}\hat{y}_r^k$ and falls in the interval $[1, 2)$. We select the set U that has an outflow closest to 1.5. The two branches are created by adding $\sum_{k\in K}\sum_{r\in R_k}\sum_{i\in U}\sum_{j\in V\setminus U} a_{rij}y_r^k = 1$ and $\sum_{k\in K}\sum_{r\in R_k}\sum_{i\in U}\sum_{j\in V\setminus U} a_{rij}y_r^k = 2$. This type of branching is equivalent to branching on the edge flow, i.e., if $U = i, j$, then the first decision imposes of a flow of 1 on the edge linking i and j (i.e., pair of arcs (i, j) and (j, i) if they both exist) and the second imposes a flow of 0 on that edge. With this branching strategy, the delivery triangle inequality can be met by modifying the adjusted arc costs.

4.6 Computational results

The proposed BPC algorithm was coded in Java and the restricted master problems were solved using version 12.4 of the linear programming solver CPLEX. The computational experiments were performed on a laptop with 2 CPU cores (clocked at 2.0 GHz) and 2 GB of RAM.

In this section, we present the computational results obtained for real-life instances provided by our industrial partner GIRO. Subsection 4.6.1 describes the test instances, Subsection 4.6.2 presents linear relaxation results, while Subsection 4.6.3 reports the integer solution results and discusses some characteristics of the computed solutions.

4.6.1 Instance description

Table 4.1 presents the test specifications of real-world problems used to evaluate the performance of the proposed algorithm, including the number of requests, the total number of vehicles, the number of contracts, and the types of vehicles. The details related to the

largest instance (S11) are presented to clarify the various aspects of the problem. In this test instance, there are 849 requests to service and two vehicle types (Taxi and Omnibus) that are governed by 5 contracts. The contract, vehicle, and break pattern specifications are provided in Tables 5.2, 5.3, and 5.4, respectively. Four of the five contracts are for Omnibus vehicles, the other involves Taxi vehicles. Only two contracts impose break requirements, while deadhead and pull-in/out costs are incurred for three contracts and a single contract, respectively.

From Table 5.3, we observe that the Taxi vehicles have a slightly higher average speed than the Omnibus vehicles, yielding faster traveling times. The service time for a request is computed as the sum of a fixed setup time and a loading/unloading time for each passenger in the request. The latter depends on the passenger type (ambulant or wheelchair). Note that, when a vehicle serves several requests together at the same location, a single setup time is needed. This is taken into account in the labeling algorithm by adjusting dynamically the service times of these requests, except the first one. The capacity of an Omnibus vehicle is much larger than that of a Taxi vehicle.

Table 5.4 indicates the characteristics of the break patterns for contracts 2 and 3, as well as the corresponding maximum route duration. There are four possible break patterns for contract 2 and three for contract 3. In each break pattern ℓ, the duration of each of its n^ℓ breaks is the same and the duration of each route segment must fall within the same interval $[\Delta^{min,\ell}, \Delta^{max,\ell}]$. Notice that the maximum route duration $T^{max,\ell}$ is not constraining for pattern 1 for both contracts, but the *wait* and *fwSl* components of the labels are required to find a feasible schedule respecting the minimum and maximum segment durations.

In this case study, the 849 requests involve 649 ambulant and 209 wheelchair passengers. Thus, some requests have more than one passenger and can combine the two types. The average width of the time windows is 20 minutes for the pickups and 50 minutes for the deliveries. To determine the maximum ride time L_i of each request $i \in P$, we first divide the requests into three categories: short distance, medium distance, and long distance. Then, L_i is equal to its direct travel time multiplied by a factor 2, 1.8, or 1.35 if i is a short, medium, or long-distance request, respectively.

The distribution of the requests' start times and their pick-up and drop-off locations are illustrated in Figures 4.1 and 4.2, respectively. We observe that the requests are spread throughout the day, with a larger number of requests starting between 10am and 9pm. Given that the number of pickup and drop-off locations is much less than the number of requests, we deduce that several requests have the same origin or the same destination, but not necessarily the same time window. Each test problem incorporates real-world details

Table 4.1 Instance specifications

Instance	Requests	Vehicles	Contracts	Types of vehicle
S1	300	25	9	3
S2	330	27	9	3
S3	350	29	9	3
S4	400	33	9	3
S5	500	42	9	3
S6	550	48	9	3
S7	600	52	9	3
S8	700	58	9	3
S9	800	62	9	3
S10	820	65	9	3
S11	849	81	5	2

Table 4.2 Contract specifications

Contract k	Vehicle type	n_k	Breaks	w_k^{Dh}	w_k^{Pull}	F_k (\$)
1	Omnibus	25	No	0	0	5,000
2	Omnibus	20	Yes	1	0	10,000
3	Omnibus	20	Yes	1	1	10,000
4	Taxi	6	No	0	0	10,000
5	Omnibus	10	No	1	0	5,000

that contribute to the overall complexity and realism of the problem.

4.6.2 Linear relaxation results

We first present linear relaxation results that allow to compare the performance of the CG algorithm with and without using the first labeling heuristic for solving the subproblems (see Section 4.5.1). These results are reported in Table 4.5. For each algorithm variant, it indicates the total CPU time required to solve only the linear relaxation (before adding cuts) as well as the total number of CG iterations (Total) and the number of iterations in which columns were generated using the first labeling heuristic (FirstLH), the second labeling heuristic (SecondLH), and the exact labeling heuristic (ExactL). From these results, we observe that using the first heuristic labeling leads to a significant average time reduction of 55% (e.g., 1059 seconds versus 2579 seconds for S11). In fact, this heuristic, which takes less time, succeeds in generating columns in 95% of iterations on average (e.g., 132 iterations out of 140 in S11). We can also notice that the exact labeling algorithm is called only in the

Table 4.3 Vehicle specifications

Vehicle	Speed (km/h)	Setup time (min)	(Un)loading time (min)		Capacity	
			Ambulant	Wheelchair	Seats	Wheelchairs
Omnibus	30	1	1	1	15	10
Taxi	33	1	1	2	6	3

Table 4.4 Break pattern specifications and maximum route durations

Contract k	Pattern ℓ	n^ℓ	δ^ℓ (min)	$\Delta^{min,\ell}$ (min)	$\Delta^{max,\ell}$ (min)	$T^{max,\ell}$
2	1	1	15	60	120	300
	2	2	15	90	180	420
	3	1	45	120	240	420
	4	3	15	90	240	840
3	1	1	15	60	120	300
	2	2	15	90	180	420
	3	3	15	90	240	840

last CG iteration and, thus, cannot generate negative reduced cost columns. It means that the second labeling heuristic, with a relaxed dominance condition on the open requests, does not seem to be very imprecise. In fact, throughout the whole search tree, the exact labeling algorithm was never called more than twice per node.

Adding cuts at the root node increases the total CPU time for both algorithms, reducing the speed by 51.8% (e.g., 1237 seconds and 2833 seconds for S11). When the first labeling heuristic was applied, a total of 361 RCIs and 59 SRIs were generated at the root node for S11.

4.6.3 Integer solution results

Table 4.6 presents the results obtained by the complete BPC algorithm (with the first labeling heuristic). In order, it specifies the total CPU time in minutes, the time spent solving the restricted master problems (RMP), the time spent solving the subproblems (SP), the number of nodes explored in the branch-and-bound (BB) search tree, the total number of RCIs and SRIs generated in the whole search tree, and the root node integrality gap in percentage (i.e., the relative difference between the optimal value and the lower bound achieved at the root node after adding cuts). These results demonstrate that our BPC algorithm can optimally solve large-scale practical DARP instances with more than 300 requests. As shown, the

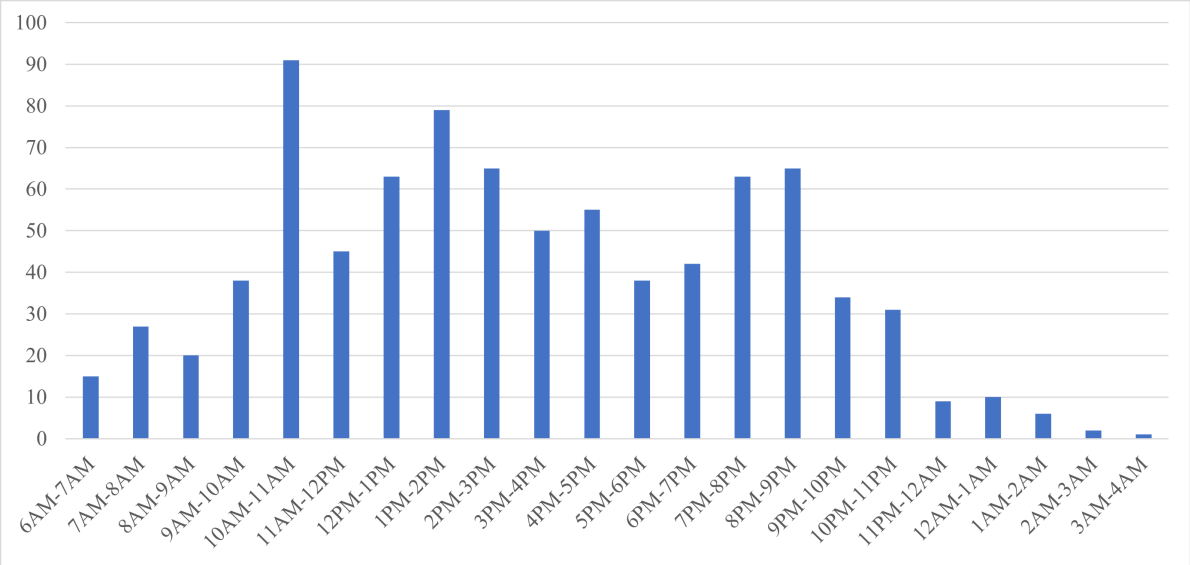


Figure 4.1 Distribution of the requests' start times

average time to solve these instances is 305 minutes, and the largest problem (S11), which involves 849 requests, achieved the optimal solution in 486 minutes (8 hours and 6 minutes). This computational time is almost evenly split between the restricted master problem and the subproblems. About S11, the number of branching nodes explored (1400) is quite large and ensues from a relatively large root node gap of 11%. The cuts added at the root node were not sufficient to yield a smaller gap (the integrality gap before adding cuts was 12.3%). Nevertheless, cuts were often generated in the search tree, especially RCIs, and were surely helpful to reduce its size.

To the best of our knowledge, these instances are by far the largest DARP instances reported to be solved to optimality in the literature. It is difficult to say what makes it possible to solve these large instances optimally in less than 8 hours. Some features like not considering costs for pull-in/out and deadhead trips for some contracts may simplify the complexity of this instance. On the other hand, we observe in the computed solution of S11 that the average number of requests per vehicle used is 17.2 and that up to 13 (resp. 6) requests can be simultaneously onboard an Omnibus (resp. Taxi) vehicle, showing that it is not a trivial instance and that the proposed state-of-the-art BPC algorithm is efficient to deal with complex practical features.

The largest instance (S11) was also solved by the heuristic algorithm commercialized by GIRO, a local-search-based metaheuristic that we cannot describe for confidentiality reasons. This heuristic took less than 5 minutes to compute its solution. Figures 4.3 to 4.5 provide information for comparing the solutions computed by this heuristic and the proposed BPC

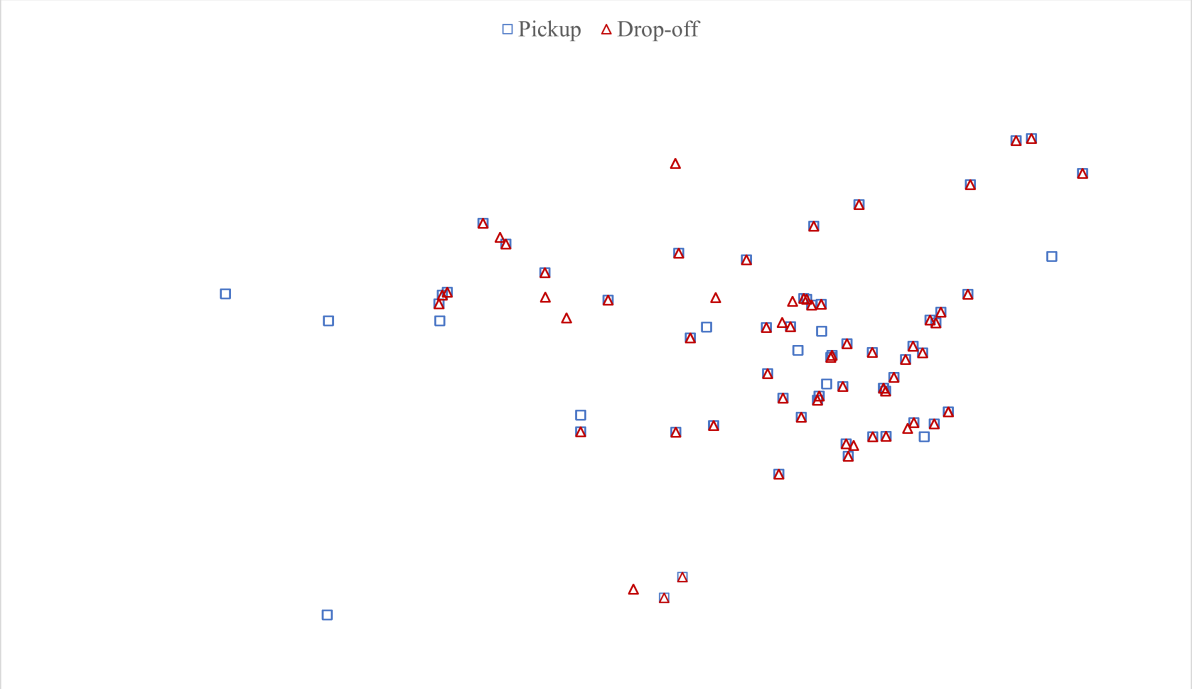


Figure 4.2 Pickup and delivery locations of the requests

algorithm. Figure 4.3 reports the total cost and the cost per contract for both solutions. Observe first that, in both solutions, contract 3 is not used at all. This can be explained by the fact that using vehicles from this contract is more expensive than from others (pull-in/out and deadhead trips have to be paid) and the number of available vehicles from the other contracts is sufficient to cover all requests. Observe also that the GIRO heuristic computes a solution with a cost that is only 0.39% larger than the optimal value. This figure also highlight that the budget allocated for each contract (see Table 5.2) is constraining only for contract 1 that is the most flexible (without breaks and maximum route duration) and the least costly (no costs for deadhead and pull-in/out trips).

Figure 4.4 shows that a total of 43 Omnibus and 6 Taxi vehicles are used in the BPC optimal solution whereas the GIRO solution requires one more Omnibus vehicle and the same number of Taxi vehicles. Finally, Figure 4.5 indicates that, between both solutions, the number of requests per contract is relatively stable for contracts 4 and 5, but varies for contracts 1 and 2. In fact, less requests are serviced by contract 2 in the optimal solution, allowing to save one vehicle. Looking only at the BPC optimal solution, we compute average numbers of requests per used vehicle equal to 18.4, 8.1, 30.0, and 23.8 for contracts 1, 2, 4, and 5, respectively. The vehicles of contract 2 are, thus, underused compared to the others, as they are the costlier. The others cannot be further exploited because of either their allocated

Table 4.5 Linear relaxation results

	With first labeling heuristic					Without first labeling heuristic			
	CPU time	Number of CG iterations				CPU time	Number of CG iterations		
Instance	(sec)	Total	FirstLH	SecondLH	ExactL	(sec)	Total	SecondLH	ExactL
S1	298	97	93	3	0	672	80	79	0
S2	346	102	98	3	0	764	83	82	0
S3	349	96	91	4	0	765	78	77	0
S4	467	108	102	5	0	1036	93	92	0
S5	628	115	108	6	0	1364	97	96	0
S6	653	111	104	6	0	1370	88	87	0
S7	711	113	107	5	0	1510	90	89	0
S8	893	124	118	5	0	1995	102	101	0
S9	978	122	116	5	0	2284	101	100	0
S10	1010	132	124	7	0	2403	108	107	0
S11	1059	140	132	7	0	2579	121	120	0

budget (contract 1) or their availability (contracts 4 and 5).

4.7 Conclusion

In this paper, we have investigated a practical DARP variant with heterogeneous customers and vehicles as well as other practical considerations (break requirements, maximum route duration, contract-based cost structure). To solve it, we have developed a state-of-the-art exact BPC algorithm. The algorithm was tested on real-world instances with 300 to 849 requests and a fleet of 25 to more than 70 vehicles. The largest instance reached the optimal solution in 8 hours and 6 minutes. To our knowledge, this DARP instance is by far the largest one reported to be solved in the literature. It was helpful, among others, to assess the quality of the solution produced by the heuristic of our industrial partner. For future studies, we aim at developing a matheuristic, possibly embedding a heuristic BPC component, for tackling much larger DARP instances (with 10,000 requests or more) faced by GIRO.

Table 4.6 Integer solution results

Insrance	CPU time (min)			Number of nodes	Number of cuts		Root node gap (%)
	Total	RMP	SP		RCI	SRI	
S1	134	62	72	534	343	120	4.3
S2	160	74	87	560	400	152	4.5
S3	178	83	96	576	371	163	4.6
S4	237	110	128	740	699	257	5.9
S5	259	119	140	796	687	312	6.3
S6	336	156	181	986	1063	419	7.8
S7	318	147	172	966	1041	398	7.6
S8	391	181	210	1154	1462	509	9.1
S9	416	193	224	1132	1428	554	8.9
S10	435	206	229	1196	1705	617	9.5
S11	486	232	255	1400	2531	1013	11.0

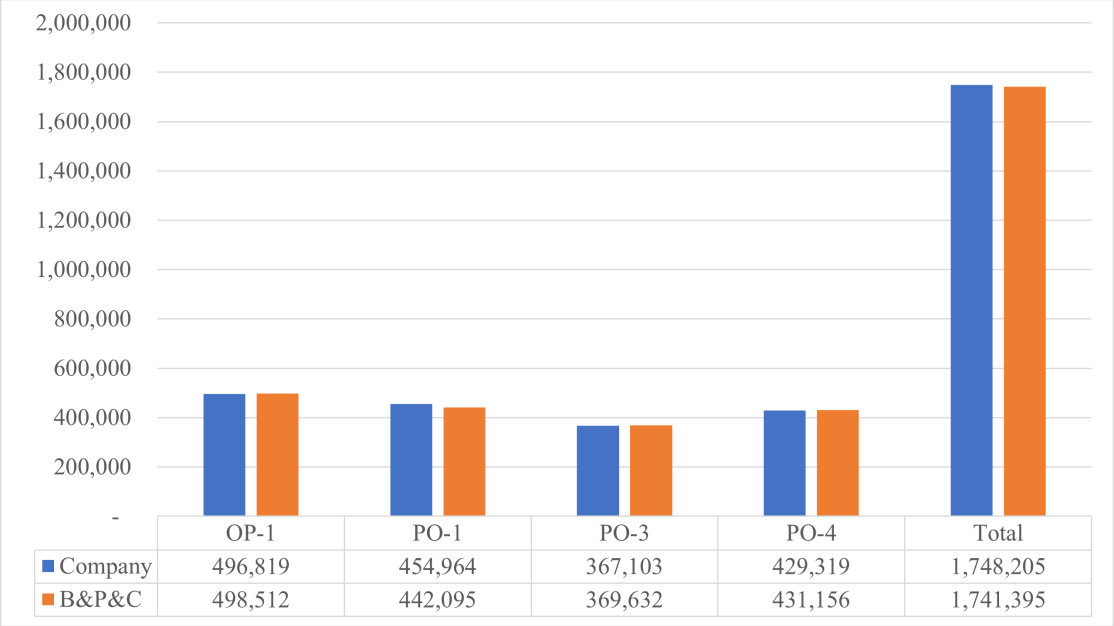


Figure 4.3 Total cost and cost per contract for both solutions

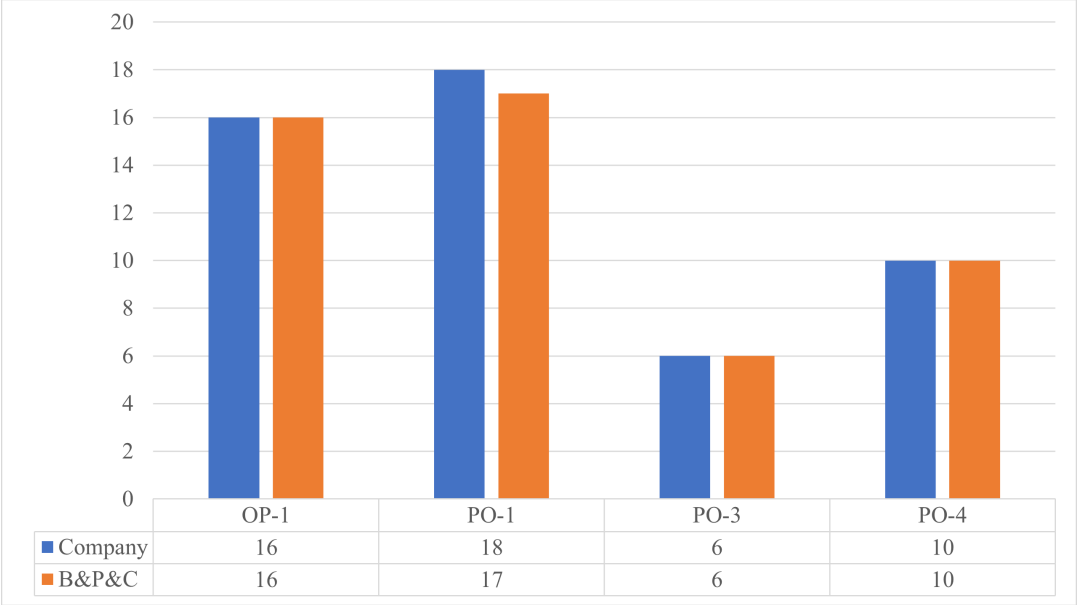


Figure 4.4 Number of vehicles used per contract in both solutions

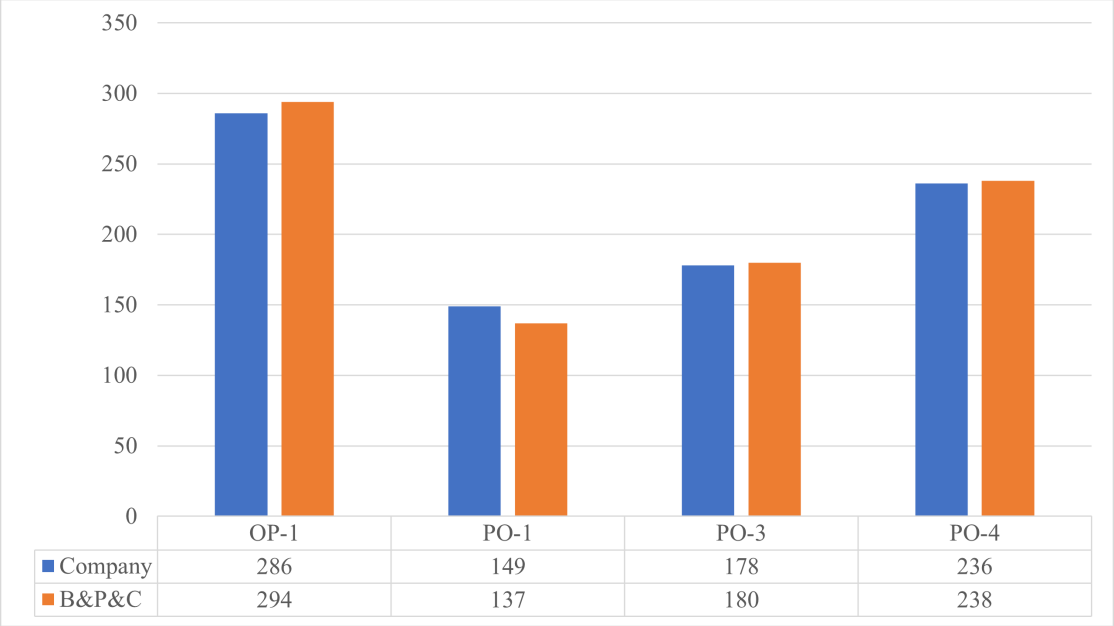


Figure 4.5 Number of requests covered by contract in both solutions

CHAPTER 5 ARTICLE 2: A VARIABLE NEIGHBORHOOD SEARCH ALGORITHM FOR A VERY LARGE-SCALE PRACTICAL DIAL-A-RIDE PROBLEM

Authors: Mohammad Karimi, Guy Desaulniers, Michel Gendreau

Note: Submitted the Computers & Operations Research on June 9, 2025

5.1 Introduction

The rapid growth of urban areas and the increasing complexity of transportation systems demand more efficient and scalable solutions to manage passenger mobility. The dial-a-ride problem (DARP) constitutes a critical optimization challenge that involves designing vehicle routes to serve a set of transportation requests, adhering to time windows and capacity constraints [4, 102]. This problem is especially relevant in scenarios like paratransit services, ride-sharing platforms, and on-demand public transportation [5–7]. The time window constraints, passenger heterogeneity (e.g., ambulant versus wheelchair passengers), and fleet diversity make the DARP a highly challenging problem to solve, especially for very large-scale instances.

The DARP encompasses a wide range of real-world characteristics that arise from various applications. In this paper, we address practical features based on a real-world case provided by our industrial partner, GIRO Inc., a Montreal-based software company. In this case, a *route* represents a vehicle's trip, which begins and ends at a depot, reflecting the work performed by the vehicle over a full day. Each route is constrained by a maximum allowable duration and specific break patterns dictated by vehicle contracts. Additionally, the number of vehicles available is limited, and each contract operates under a budget based on a detailed cost structure. Managing these constraints becomes particularly challenging in very large-scale instances. To address these challenges, we propose a method based on variable neighborhood search (VNS), enhanced with mixed-integer programming techniques for initial solution generation, neighborhood exploration, and local search. The algorithm was tested on very large-scale real-world instances, ranging from 2,932 to 10,527 requests, with vehicle fleets ranging from 198 to 563 vehicles. In addition, the proposed algorithm was evaluated on medium-sized problem instances comprising between 300 and 849 requests. Our method consistently produced effective solutions within short computation times.

The remainder of this paper is organized as follows. Section 5.2 reviews the related litera-

ture on the DARP and large-scale routing problems. In Section 5.3, we present a detailed description of the specific DARP variant under consideration. Then, Section 5.4 outlines the proposed algorithm in depth. Section 5.5 introduces the data used in the case study and presents our computational results. Finally, Section 5.6 concludes the paper with a summary of our findings.

5.2 Literature review

The DARP is a specific variant of vehicle routing problems (VRP) with pickups and deliveries, where a fleet of vehicles transports passengers between pickup and delivery points. The literature has examined a broad range of problem characteristics. Foundational studies typically considered simplified settings with a homogeneous fleet of capacitated vehicles, pickup and delivery time windows, limits on passenger ride times, and restrictions on route duration [17,61]. Building on these basics, some works have extended the problem to incorporate features motivated by practical applications, such as heterogeneous passengers and vehicles [12,74,112,113], passenger transfers [35,36], integration with public transport services [38,47], requirements for additional manpower [23], and driver scheduling constraints [19, 41]. These extensions reflect the complexity of real-world systems and have significantly shaped algorithmic developments in the field.

Given the high computational complexity of exact methods [18], particularly for large-scale instances, various heuristic and metaheuristic approaches have been developed to provide more efficient solutions [11, 14, 33, 35, 40, 69, 76]. While significant progress has been made in solving small- to medium-sized DARP instances, few efforts have dealt with large-scale instances very common in large cities or highly-populated regions. Of the few studies that have addressed large-scale instances of DARP, Xiang et al. [77] introduced a heuristic framework incorporating local search, diversification, and intensification strategies. Their approach was tested on instances with up to 2,000 requests, yielding solutions in acceptable computational times, demonstrating the potential of organized search methods to handle large-sized problems efficiently.

Muelas et al. [78] presented a distributed VNS algorithm based on Muelas et al. [79]. By partitioning the request space and integrating route combinations, their method addressed instances with up to 16,000 requests, as tested in the city of San Francisco. The distributed nature of their approach, along with effective route partitioning, allowed for the resolution of high-dimensional problems in computationally feasible time frames. Their results represent a significant step forward in addressing the scalability of DARP solutions in real-world scenarios. Recently, Liu et al. [80] investigated the use of shared autonomous vehicles to

provide dial-a-ride services in urban and rural settings, focusing on large-scale homogeneous DARP. They proposed a greedy insertion heuristic, enhanced by a filtering system to accelerate decision-making, and an innovative network-driven route encoding. Their approach was tested on instances ranging from 10,000 to 300,000 requests across networks with 1,000 to 15,000 nodes, demonstrating its scalability and efficiency in managing large-scale DARP instances.

Parallel developments in the VRP have also sought to extend solution methodologies to large-scale instances. Kytöjoki et al. [81] proposed a VNS-based approach for solving very large-scale VRP instances. Their method, which combines various insertion heuristics to construct initial solutions and subsequently improves them through a variable neighborhood descent, demonstrated the capability to solve instances with up to 20,000 customers within an hour. This work underscores the efficacy of hybridized heuristic strategies in tackling large-scale problems and serves as an important reference for scalable DARP solutions. Further contributions to the VRP literature include the work of Arnold et al. [83], who developed a local search heuristic capable of solving capacitated VRP instances with up to 30,000 customers. By leveraging pruning techniques and sequential search strategies, their algorithm produced high-quality solutions in a reasonable time. Similarly, Accorsi & Vigo [84] advanced the field with the FILO algorithm, a hybrid iterated local search that combines acceleration techniques with a simulated annealing-based acceptance criterion. Their method exhibited a strong ability to navigate large search spaces, yielding competitive results on very large VRP instances within 150 minutes. Recently, Máximo et al. [86] presented an enhanced solution method for the capacitated VRP by developing an adaptive iterated local search algorithm in two search phases, both incorporating perturbation and local search, but differing in how reference solutions are selected. In the first phase, the reference solution is chosen using an acceptance criterion, while in the second phase, it is selected from an elite set of the best solutions found. This algorithm, termed AILS-II, outperforms existing methods on smaller instances and proves particularly effective on large-scale problems with up to 30,000 vertices, achieving superior solution quality.

Despite these advances in the VRP and DARP literature, few works have explicitly addressed the unique challenges posed by very large-scale practical DARP instances. The complexity of such problems, characterized by high-dimensional search spaces and computationally expensive constraints, necessitates the development of advanced algorithms capable of balancing solution quality and computational efficiency. Our proposed VNS algorithm contributes to this ongoing research by introducing a novel approach specifically designed to solve very large-scale practical DARP instances. Building upon existing VNS frameworks, we incorporate probabilistic neighborhood selection mechanisms and enhanced shaking procedures to

improve solution diversity and quality. These innovations are tailored to the demands of large urban transportation networks, where the ability to generate high-quality solutions within limited computational time is critical. By addressing the gap in scalable DARP solutions, our work aims to push the boundaries of what is computationally feasible in solving large-scale transportation problems.

5.3 Problem definition

In this paper, we tackle the variant of the DARP introduced in our previous work [114], which explicitly models heterogeneous transportation requests: passengers may be ambulant or in a wheelchair. Both passenger types require a seat, but wheelchair passengers additionally require space for their wheelchairs. This distinction also impacts the service time required for each request. Moreover, each transportation request is subject to time window constraints at both the pickup and drop-off locations, along with a maximum allowable ride time for passengers. The available vehicles operate under different contractual agreements, with each contract specifying the number of vehicles, a cost structure, and a maximum budget for each contract. These contracts define specific vehicle types with varying capacities. A vehicle's route is defined as a sequence of activities, including departing from the depot (pull-out), performing pickups and drop-offs, taking mandatory breaks, and returning to the depot (pull-in). The route can be segmented into one or more blocks, where a block represents a part of the route between two consecutive stops where the vehicle is empty. Contracts impose additional constraints, such as maximum route duration and mandatory break patterns. These break patterns depend on the route's total duration and specify the number of breaks required, as well as the permissible intervals between breaks or between the start and end of the route. Different vehicle types also introduce variability in travel times and distances, which in turn affects the cost structure and overall scheduling. Given these characteristics, the DARP variant presented in this paper introduces significant operational challenges, particularly when dealing with large-scale instances, heterogeneous passenger types, and complex contract-specific constraints. Addressing these challenges efficiently is key to producing practical and cost-effective solutions.

This practical DARP is defined on a directed graph $G = (V, A)$, where V is the set of vertices and A is the set of arcs. The vertex set V consists of two depots (denoted as vertices 0 and $2n + 1$), a set of pickup vertices $P = \{1, \ldots, n\}$, and a set of delivery vertices $D = \{n + 1, \ldots, 2n\}$. Each pickup vertex $i \in P$ corresponds to a request to transport one or more passengers from pickup location i to the corresponding delivery location $n + i$. These passengers may require one of two transportation modes: ambulant or wheelchair.

When a passenger boards a vehicle, they consume resources. Let $M = \{St, Wc\}$ represent the set of resources, where St denotes the number of seats and Wc denotes space for a wheelchair. For each vertex i, let q_i^m represent the quantity of resource $m \in M$ consumed at that vertex. Specifically, $q_i^m \geq 0$ if $i \in P$ (pickup vertices) and $q_{n+i}^m = -q_i^m \leq 0$ if $n + i \in D$ (delivery vertices), with $q_i^m = 0$ at any other vertex. In addition, each request $i \in P$ must be completed within a maximum allowable ride time L_i and each vertex $i \in V$ has an associated time window $[e_i, u_i]$, where e_i is the earliest time and u_i is the latest time that service can begin at vertex i. The extremities of these time windows for the depots (vertices 0 and $2n+1$) represent the overall planning horizon.

The use of vehicles in this problem is governed by a set of contracts K. Each contract $k \in K$ specifies a fleet of n_k identical vehicles, a total operating budget F_k, a cost structure, and a set of operational rules. Contract k defines a vehicle's capacity Q_k^m for each resource $m \in M$, and the service time s_i^k at each pickup and delivery vertex $i \in P \cup D$. The travel time between any two vertices $i, j \in V$ also depends on the contract, based on the vehicle's speed, and is denoted by t_{ij}^k. Likewise, the cost of traveling between two vertices depends on both the contract and other conditions, which will be further described. We assume that all travel times and costs satisfy the triangle inequality, ensuring that direct travel between two locations is never more expensive or time-consuming than traveling via an intermediate point.

A route for a contract $k \in K$ is defined as a path from the starting depot 0 to the ending depot $2n + 1$, accompanied by a schedule that specifies the service times at each visited vertex. A route is feasible if it satisfies several key constraints. First, the route must respect the pairing and precedence of requests: if a pickup vertex $i \in P$ is visited, the corresponding delivery vertex $n+i$ must also be visited afterward, but no later than L_i minutes after the service at i begins. Additionally, each service at a vertex must occur within its designated time window. The route must also always comply with vehicle capacity limits for the available resources. For certain contracts, routes may be subject to maximum duration constraints and must account for required breaks, defined by specific break patterns. A break pattern ℓ depends on the total route duration and specifies the maximum route duration $T^{max,\ell}$, the number of breaks n^ℓ (up to three in this case), and the duration d_b^ℓ of each break $b = 1, \ldots, n^\ell$. The breaks divide the route into $n^\ell + 1$ segments, where the duration of each segment must fall within a given range $[\Delta_g^{min,\ell}, \Delta_g^{max,\ell}]$. Importantly, breaks can only take place when there are no passengers onboard, typically occurring after the last passenger has been dropped off. It is also assumed that the travel time between any two locations is always less than the minimum segment duration $\Delta_g^{min,\ell}$, ensuring that at most one break can be scheduled along any arc in the route. Breaks are not allowed on the pull-out or pull-in arcs of a route, maintaining the

continuity of the vehicle's operation throughout the day.

An illustrative example of such a route is shown in Figure 5.1. This route features a break pattern with three breaks, dividing the route into four segments. Each segment contains one or more service blocks, composed of consecutive pickups and drop-offs, and their durations satisfy the prescribed interval constraints $[\Delta_g^{min}, \Delta_g^{max}]$. The three breaks, each of fixed duration d_b, contribute to the total route time, which must remain below the maximum allowable duration T^{max}. Breaks are scheduled only after all passengers of the corresponding segment have been dropped off, thereby ensuring feasibility with respect to operational rules. This example highlights how both the segmentation and the overall route duration are jointly constrained by the break pattern.

The cost of a route c_r^k depends on the contract $k \in K$ and is influenced by two binary parameters: w_k^{Pull} and w_k^{Dh}. These parameters indicate whether pull-ins/outs and deadheads (empty vehicle travel between blocks) incur costs. Specifically, w_k^{Pull} determines if the pull-in and pull-out movements are costed, while w_k^{Dh} governs whether deadhead trips between blocks are charged. In addition, the number of blocks in a route is equal to the number of deadheads plus one (pull-out). The total cost of the route is then calculated based on these conditions, reflecting the contract's cost structure for vehicle operations. It is defined by:

$$\begin{aligned} c_r^k &= \gamma^{Blk} m_r^{Blk} + \gamma^{Dur}(\eta_r^{Pax} + \eta_r^{Brk} + w_k^{Pull}\eta_r^{Pull} + w_k^{Dh}\eta_r^{Dh}) \\ &\quad + \gamma^{Dst}(\chi_r^{Pax} + w_k^{Pull}\chi_r^{Pull} + w_k^{Dh}\chi_r^{Dh}), \end{aligned} \tag{5.1}$$

In this cost structure, m_r^{Blk} represents the number of blocks in route r, while η_r^g denotes the total travel duration for various components of the route including the travel time with passengers onboard when $g = Pax$, the duration of mandatory breaks when $g = Brk$, the pull-in and pull-out times when $g = Pull$, and the deadhead times between blocks when $g = Dh$. Similarly, χ_r^g corresponds to the total distance for these components: the distance traveled with passengers (Pax), the pull-in/pull-out segments ($Pull$), and the deadhead segments (Dh). The unit costs for each component are given by γ^g , where $g = Blk$ for the cost per block, $g = Dur$ for the cost per time unit, and $g = Dst$ for the cost per distance unit. Penalizing the number of blocks in a route encourages the avoidance of empty travel periods, thus increasing the proportion of time that the vehicles carry passengers. Even if they may slightly increase the traveling costs, such routes are often preferred by the para-transit societies according to our industrial partner because they are more attractive to the contractors. Note also that the weight of this term in the objective function is small compared to the other terms, so its influence on route design remains limited.

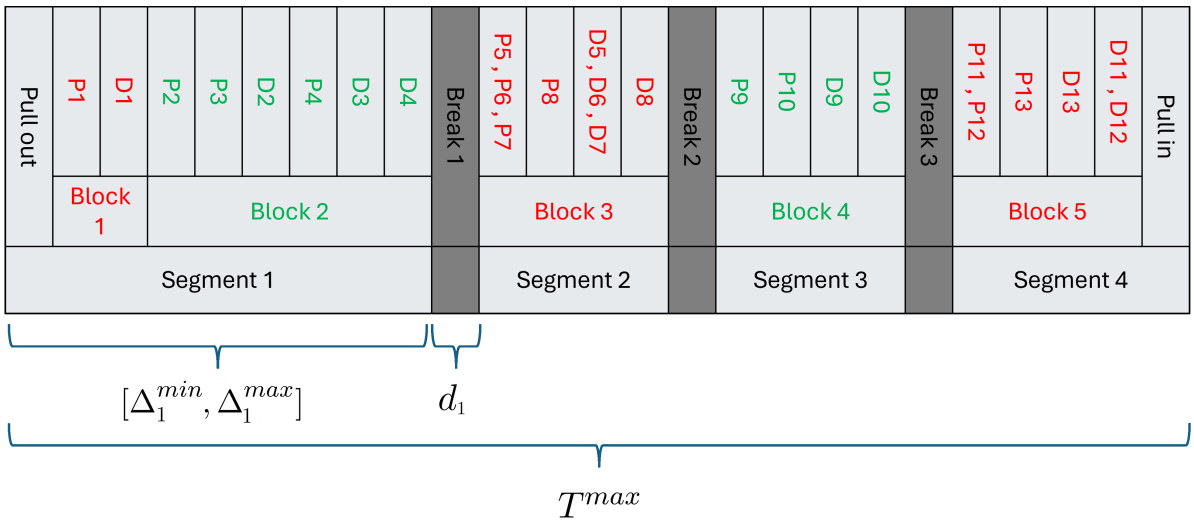


Figure 5.1 Example of a route with breaks

Given this route cost that depends on the contract k and other conditions, we define the following cost c_{ij}^k:

$$c_{ij}^k = \begin{cases} \gamma^{Blk} + w_k^{Pull}(\gamma^{Dur}\eta_{ij}^{Pull} + \gamma^{Dst}\chi_{ij}^{Pull}) & \text{if } i = 0 \\ w_k^{Pull}(\gamma^{Dur}\eta_{ij}^{Pull} + \gamma^{Dst}\chi_{ij}^{Pull}) & \text{if } j = 2n+1 \\ \gamma^{Blk} + \gamma^{Dur}\eta_{ij}^{Brk} + w_k^{Dh}(\gamma^{Dur}\eta_{ij}^{Dh} + \gamma^{Dst}\chi_{ij}^{Dh}) & \text{if } (i,j) \text{ is a deadhead} \\ & \quad \text{with a break} \\ \gamma^{Blk} + w_k^{Dh}(\gamma^{Dur}\eta_{ij}^{Dh} + \gamma^{Dst}\chi_{ij}^{Dh}) & \text{if } (i,j) \text{ is a deadhead} \\ & \quad \text{without a break} \\ \gamma^{Dur}\eta_{ij}^{Pax} + \gamma^{Dst}\chi_{ij}^{Pax} & \text{if } (i,j) \text{ is traversed with} \\ & \quad \text{onboard passengers,} \end{cases} \tag{5.2}$$

where the arc parameters η_{ij}^g and χ_{ij}^g, $g = Pax, Pull, Dh$, are the counterparts of the route parameters η_r^g and χ_r^g. The last three cases cannot be determined a priori, but rather dynamically when building routes.

5.4 The proposed algorithm

The practical DARP is addressed using a VNS heuristic. The core idea behind this algorithm is to start with an initial solution, denoted as s_{init}. This initial solution serves as the first incumbent solution and current solution s. The heuristic uses a sequence of neighborhood

classes (or shaker methods) indexed by k. In each iteration, a neighborhood class is applied to create a solution s' within the neighborhood denoted by $N_k(s)$. Next, if the solution is promising (Section 5.4.3), a local search algorithm is employed on s', leading to an improved solution s''. If s'' is better than the current incumbent s, it replaces s. On the other hand, if s'' does not improve upon s, the incumbent remains unchanged, and an acceptance criterion is used. This iterative process is repeated until a predetermined stopping criterion is met, ensuring the search thoroughly explores multiple neighborhoods for good solutions. The general framework of our VNS heuristic is given in Algorithm 1.

Algorithm 1 VNS heuristic

```
// initial solution
Generate s_init;
Set s := s_init; Set k := 1; Set s_best := s;
repeat
    // shaking
    Compute s' in neighborhood N_k(s);
    // local search
    if s' is a promising solution then
        Apply local search to s' yielding s'';
    else
        Set s'' := s';
    end if
    if s'' is better than s then
        Set s := s''; Set k := 1;
        if s'' better than s_best then
            Set s_best := s'';
        end if
    else
        Set k := k + 1;
        if acceptance criteria are satisfied then
            Set s := s'';
        end if
    end if
until some stopping criterion is met
return s_best;
```

The following subsections describe the methods used in our VNS algorithm. Section 5.4.1 explains the approach for generating the initial solution. Section 5.4.2 outlines the neighborhood searches employed to explore the solution space. Finally, Section 5.4.3 presents the local search technique to refine the generated solutions.

5.4.1 Generating the initial solution

In the literature, various methods have been proposed for generating an initial solution, particularly in high-dimensional problems like the practical DARP. Starting with a feasible solution is critical, as it provides a solid foundation for subsequent improvements during the optimization process. We propose three different methods for generating an initial solution, each of which is described in detail below.

Greedy-by-route heuristic

This method requires a pre-processing step to identify neighboring requests for each request. A neighboring request is defined as one that can be directly accessed after serving the current request within a specified time interval. In this implementation, requests that are accessible within one hour after a given request are considered neighbors and are sorted by travel duration from the end location of the first request to the start location of the second one. The method begins by sorting all requests based on their pickup times. The first request from the sorted list is selected and assigned to a vehicle that is compatible with the request in terms of time and capacity. Next, a neighboring request is chosen from the identified list of neighbors and inserted at the end of the vehicle's current route. This process continues, adding neighbors sequentially until the route is no longer feasible due to capacity, time, or other constraints. If a request cannot be inserted at the end of the route, alternative positions are considered using the best insertion method [115]. If a request cannot be feasibly inserted in the current route, even after applying the best insertion method, the algorithm attempts to insert it into another existing route. If no feasible insertion is found in any existing route, a new route is initiated for that request. This ensures that all requests are eventually assigned to routes. Once the route is constructed and feasible, all requests included in the route are removed from the request list. The algorithm then selects the next request from the sorted list, assigns it to a compatible vehicle, and repeats the process until all requests are assigned to routes. In this method, the routes primarily follow a sequence of servicing individual requests (pickup and drop-off), making it less likely for a vehicle to handle multiple requests simultaneously, unlike the next two methods.

Sliding window heuristic

First, all requests are sorted based on their pickup times. The day is then divided into hourly periods. The requests within the first period are inserted into routes using the best insertion method. If a request cannot be inserted into existing routes, a new route is started

to accommodate it, ensuring that all requests are eventually assigned. Once all requests from the first period have been assigned to routes, a limited version of the VNS algorithm is applied to improve the current routes' quality. "Limited" VNS refers to the use of only small-size neighborhoods (specifically, chain and swap shakers of sizes 1 and 2) which allows for local route improvement while keeping the search effort bounded. After optimizing the routes for the first period, the requests from the next period are inserted into the same routes, continuing the process. The algorithm iterates in this manner—optimizing routes after each period—until all requests are assigned to feasible routes for the entire day.

LP-based heuristic

As shown in Algorithm 2, the approach begins by dividing the day into discrete hourly periods, ensuring the problem is tackled incrementally across manageable time intervals. For each period starting with the earliest, a DARP model is formulated based on [12] (see Appendix A), incorporating the requests specific to that period along with the fleet of available vehicles. This arc-flow model involves the binary variables x_{ij} indicating the vehicle flow on each arc (i, j). To mitigate the computational burden, the linear relaxation of the model is first solved. The binary variables that take non-zero values in the relaxation are retained, while their costs are adjusted using the formula: $c''_{ij} = \frac{c'_{ij}}{\bar{x}_{ij}}$, where $\bar{x}_{ij}$ represents the relaxed variable values, c''_{ij} and c'_{ij} are the new and current costs, respectively. This cost-adjusted linear relaxation model is then resolved iteratively, refining the variable selection over several cycles. After a predetermined number of iterations ($MaxIteration$), any binary variables that consistently remain inactive (i.e., zero) are fixed, effectively pruning the solution space. The reduced model, now significantly smaller, is subsequently solved as an integer program within a specified time limit, leveraging the remaining active variables for optimization. This process is repeated for each subsequent period, taking into account the status of vehicles already in service and the availability of additional vehicles, ensuring that each period is handled in sequence while managing the computational complexity effectively.

Algorithm 2 LP-based heuristic for initial solution generation

```
Divide the length of the day into periods;
Set per := 1;
repeat
    Set iteration := 1;
    Set c'_ij := c_ij;
    Set c''_ij := c_ij;
    Formulate a DARP model for requests and vehicles in period per
    Set ActiveVariables := ∅;
    repeat
        Solve linear relaxation of the model with cost coefficients c'_ij;
        Set ActiveVariables := ActiveVariables ∪ {(i, j) : x̄_ij > 0};
        Set c''_ij := c'_ij / x̄_ij for (i, j) ∈ {(i, j) : x̄_ij > 0};
        Set c'_ij := c''_ij;
        Set iteration := iteration + 1;
    until iteration ≤ MaxIteration
    Fix inactive variables ((i, j) ∉ ActiveVariables) to zero;
    Solve integer programming model with ActiveVariables in a limited time;
    Set per := per + 1;
until per ≤ TotalPeriods
return initial solution;
```

5.4.2 Neighborhood classes

Five different neighborhood classes (or shakers) are used in the algorithm, with an IP-based neighborhood search method specifically designed for this problem. Most of these shakers can be parameterized by a size value, which controls the maximum number of requests or routes that can be modified during each application. The details of these shakers are provided below.

Swap neighborhood (S): This shaker, proposed in [31], performs an exchange of requests between two different routes. Initially, two distinct routes are selected using a roulette wheel selection procedure (Section 5.4.2), ensuring a probabilistic and diverse choice. For each chosen route, a sequence of requests to be swapped is then randomly determined. This involves selecting both the starting vertex of the sequence and its length, with the maximum allowable length defining the size of the neighborhood. Importantly, for every request within the selected sequence, the corresponding origin or destination vertex must also be included in the swap, even if it lies outside the chosen sequence. Once the sequences are defined, all requests in the respective sequences are removed from their current routes. These requests are then reinserted, one by one, into the alternate route, ensuring that feasibility is maintained

throughout the process.

Chain neighborhood (C): The second neighborhood class, also defined in [31], employs the ejection chain idea. Initially, a sequence of vertices is randomly chosen, similar to the swap neighborhood, and moved from one route to a second route. The selection of the second route follows a probabilistic approach using the roulette wheel mechanism of Section 5.4.2, ensuring diverse and dynamic choices. A random sequence length l is then selected, with l representing the maximum sequence length allowed by the shaker size. Next, from the second route, a sequence of length l is selected, again using the roulette wheel procedure, and moved to a third, randomly chosen route. This step is repeated until the maximum number of sequence moves, defined by the shaker's size, has been reached. This size determines both the maximum number of sequences that can be moved and the maximum length of the sequences selected.

IP-based neighborhood: The shaker starts by selecting two different routes using the roulette wheel selection procedure. Afterward, based on the hourly division of the day, the period with the highest number of requests shared between the two routes is identified. Importantly, each route must have at least one request during the selected period. Once the time period is determined, a DARP model, as shown in Appendix A, is built for the two routes, where only the requests within the selected period remain flexible, while the rest of the routes outside this period are fixed. The objective is to find the optimal reassignment of requests between the two routes for the given period. Requests are included based on their pickup times. If a delivery from a previous period occurs in the current period, it is fixed, but if a request is picked up during the selected period and delivered in the next, it remains part of the model. This DARP model is then solved using a mixed-integer programming solver, optimizing the configuration of the requests between the two routes for that period while maintaining overall feasibility. For our tests, the neighborhood size (i.e., the number of requests that remain flexible in the IP model) does not exceed eight requests, which keeps the resulting IP subproblems computationally tractable.

All natural sequences combinations neighborhood (Z): This neighborhood class leverages the concept of natural sequences, initially developed for the zero-split neighborhood [31]. A natural sequence consists of a set of vertices such that the vehicle load returns to zero at the end of the sequence. In this shaker, introduced in [79], each natural sequence is treated as a single unit, and all possible swaps between pairs of natural sequences across different routes are evaluated. For example, if two routes, i and j, contain each two natural sequences, the evaluation of the pair (i, j) involves computing the possible swap values for all four potential combinations of these sequences. This ensures a comprehensive exploration of swapping

possibilities between routes. This neighborhood class does not have a size parameter, as it always performs the same set of operations, focusing on full natural sequence swaps to explore the solution space efficiently.

Exchange-vehicle (E): In this shaker, the vehicles assigned to a pair of routes are swapped. The selected vehicles come from different contracts, ensuring that the change in the cost structure is taken into account when evaluating potential improvements in the solution. By swapping vehicles between routes with varying contractual terms, the shaker captures the impact of operational cost differences, which can lead to a more cost-effective and optimized solution overall. This approach adds a layer of complexity and realism by addressing contractual heterogeneity in vehicle assignments.

The described shakers may be applied in various orders and with different neighborhood sizes, allowing for flexibility in their implementation. However, in adherence to the guiding principles of the VNS algorithm, it is advisable to apply the shakers that introduce fewer perturbations at the outset. This approach reflects the fundamental strategy of VNS, where smaller, less disruptive changes are explored initially to retain promising solution features, followed by larger perturbations that progressively diversify the search. By following this hierarchical sequence, the search process remains both efficient and focused, improving the chances of identifying high-quality solutions without unnecessary computational overhead.

Roulette wheel route selection

To guide the application of shakers, a roulette wheel selection method was combined with the route stability idea [13]. In this approach, each route is assigned a stability counter, which tracks the number of iterations the route has remained unmodified. Routes with higher stability counters are prioritized for shaking, as they may be stuck in local optima. To implement this, the stability counters are used to determine the probability of selecting a route for modification. Specifically, the stability counters are summed to calculate a total stability score, and each route's probability of being selected is proportional to its stability counter relative to the total score. This way, routes that have not been modified for longer periods have a greater chance of being selected, but all routes still maintain some probability of being chosen, ensuring diversity in the search process. The roulette wheel selection works by generating a random number between zero and the total stability score. The algorithm then iterates through the routes, accumulating their stability counters until the random number falls within the range corresponding to a particular route. This route is then selected for a shaker application. After applying a shaker, the stability counter of the chosen route is reset to zero, while the counters of all other routes are incremented, reflecting their continued

stability. This dynamic system ensures that routes that remain unmodified over multiple iterations become more likely to be selected, while still allowing occasional selection of less stable routes. This method provides a balanced way to prioritize the exploration of stable routes while maintaining some level of probabilistic selection to avoid local optima.

5.4.3 Local search

While the previous neighborhood classes emphasize inter-route modifications, the local search methods focus on intra-route optimization by greedily improving the sequence of vertices within each route. Given the time-consuming nature of local search, it is not applied after every shaking step. Instead, it is reserved for *promising* solutions — those that have the potential to yield new incumbent solutions. After several experiments, a solution s' is considered *promising* if its cost does not exceed by more than 1% the cost of the current solution s. There are various local search algorithms tailored to DARP, each offering unique strategies for optimizing routes. These approaches are detailed below and aim to refine routes by improving timing, minimizing travel distance, and better utilizing vehicle capacity, ultimately seeking to enhance overall solution quality efficiently.

IP-based method

In this local search algorithm, the DARP model is constructed and solved for a single route, focusing on the requests within a specific period. Based on extensive testing, the most effective approach for selecting the period is to choose the interval with the highest number of requests and include its two adjacent periods. This method ensures that a more comprehensive set of requests is considered, allowing for better optimization across neighboring time slots and improving the overall routing efficiency.

Route adjustment method

This local search method, based on [31], is systematically applied to each route as follows: initially, both the first pickup vertex and its corresponding delivery vertex are removed from the route. Subsequently, the pickup vertex is reinserted at the earliest feasible position that complies with the designated time window constraints. Following this, the delivery vertex is inserted at the first available position that respects the timing of the newly positioned pickup vertex. If this insertion results in an improvement in the cost, the local search process advances to the next pickup vertex within the route. Conversely, if no improvement is noted, the delivery vertex is placed at the subsequent available position. This iterative approach

continues until either an improvement is achieved or all potential insertion positions for the delivery vertex have been exhausted. In scenarios where no improvements are found throughout this procedure, the pickup vertex retains its original position, and the algorithm proceeds to the next pickup vertex in the route. Once the optimization process for a particular route is completed, the same algorithm is applied to the subsequent route.

5.4.4 Break insertion

In the process of generating the initial solution for contracts that necessitate breaks, potential break locations are determined based on the break configuration and the route's duration. This involves analyzing the route structure and identifying optimal positions for the insertion of breaks, ensuring compliance with operational constraints. When modifications to the route are made—such as the removal or insertion of requests—the necessity for re-evaluating the break configuration is assessed in relation to the updated route duration. If the adjustments do not warrant a change in the break configuration, the original break locations remain unchanged. Conversely, if a change is necessary, the best potential locations for the break are evaluated to identify the most suitable position for insertion. It is crucial to note that if the insertion of a break leads to the inability to construct a feasible route, the proposed solution will be deemed infeasible and subsequently rejected.

5.4.5 Acceptance criterion

For deciding whether the incumbent solution should move to the new solution s'' or not, the algorithm applies an acceptance criterion governed by a simulated annealing approach [79]. A worse solution is accepted with a probability of $\exp\left(\frac{-\Delta E}{T}\right)$, where ΔE is the difference in the objective value between the current and new solutions, and T represents the temperature parameter. The temperature, T, is linearly decreased based on the initial temperature and the cooling rate.

5.5 Computational results

The proposed algorithm was coded in Java and used version 12.4 of the mixed-integer programming solver CPLEX. The computational experiments were performed on a laptop with 8 CPU cores (clocked at 3.2 GHz) and 16 GB of RAM. This section presents the computational results obtained for real-life instances provided by our industrial partner GIRO. Subsection 5.5.1 describes the test instances, while Subsections 5.5.2 and 5.5.3 present results and discuss some characteristics of the computed solutions for very large-scale and

medium-sized instances, respectively.

5.5.1 Instance description

In Table 5.1, we summarize the key specifications of the real-world test instances used to evaluate the proposed algorithm's performance, including the number of requests, total vehicles, contracts, and vehicle types. The first three instances (S1-S3) are derived from real-world data provided by our industrial partner, GIRO, representing Canadian cities. To ensure a thorough evaluation, two additional instances (S4 and S5) were created from the largest real instance, S3. This allows us to assess the algorithm's performance across a range of problem sizes and complexities.

Instance S3, the largest of the real-world cases, comprises 10,527 requests and three vehicle types: Sedan, AT, and Friendly. These vehicles are managed under 9 distinct contracts. The specifications for contracts, vehicles, and break patterns are detailed in Tables 5.2, 5.3, and 5.4, respectively. Of the nine contracts, four use AT vehicles, three involve Friendly vehicles, and two utilize Sedan vehicles. Notably, three contracts do not impose break requirements, while three contracts account for deadhead and pull-in/out costs. The vehicle specifications in Table 5.3 reveal that Sedan and AT vehicles have slightly higher average speeds compared to Friendly vehicles, resulting in faster travel times. The service time for each request is calculated as the sum of a fixed setup time and a loading/unloading time, both depending on whether the passenger is ambulant or uses a wheelchair. When serving multiple requests at the same location, only a single setup time is required. In terms of capacity, AT vehicles offer a lower wheelchair capacity compared to the other two types.

Table 5.4 outlines the break patterns across contracts and provides the corresponding maximum route durations. For contracts 7 and 9, three break patterns are available, while other contracts requiring breaks have a single pattern. Each break pattern ℓ includes a specific number of breaks, n^ℓ, and mandates that the duration of each segment within the route falls between $[\Delta^{min,\ell}, \Delta^{max,\ell}]$, ensuring compliance with operational regulations.

In the case study of instance S3, the 10,527 requests include 10,605 ambulant passengers and 1,926 wheelchair passengers, highlighting that some requests consist of multiple passengers, possibly combining both types. The average width of the time windows is 30 minutes for pickups and 55 minutes for deliveries, providing flexibility in scheduling.

To calculate the maximum ride time L_i for each request $i \in P$, requests are classified into three categories based on distance: short, medium, and long distance. The maximum ride time L_i is then determined by multiplying the direct travel time by a factor of 1.75 for short-

Table 5.1 Instance speciдfications

Instance	Requests	Vehicles	Contracts	Types of vehicle
S1	2932	198	8	3
S2	7753	517	9	3
S3	10527	563	9	3
S4	5200	300	9	3
S5	5327	320	9	3

Table 5.2 Contract specifications

Contract k	Vehicle type	n_k	Breaks	w_k^{Dh}	w_k^{Pull}	F_k (\$)
1	Sedan	36	No	0	0	7,000
2	AT	69	Yes	0	0	6,000
3	Sedan	49	No	0	0	7,000
4	AT	69	Yes	0	0	6,000
5	AT	67	Yes	0	0	6,000
6	AT	58	Yes	0	0	6,000
7	Friendly	165	Yes	1	1	10,000
8	Friendly	7	No	1	1	10,000
9	Friendly	43	Yes	1	1	10,000

distance, 1.7 for medium-distance, and 1.65 for long-distance requests. This categorization ensures that the travel constraints are tailored to the nature of the request, optimizing the routing process while maintaining service quality. The data also reveal that the number of unique pick-up and drop-off locations is significantly smaller than the number of requests, indicating that many requests share the same origin or destination, though they do not necessarily share the same time window. This clustering of locations, combined with the varying time windows, adds to the complexity and realism of each test problem, ensuring that the challenges encountered reflect real-world operational scenarios.

5.5.2 Result of very large-scale instances

To assess the performance of the proposed algorithm and its components, we present the results of various implementations and configurations. The following sections provide a detailed analysis of each part of the algorithm, along with tests involving different combinations of these components. This comprehensive evaluation aims to highlight each element's contribution and the algorithm's overall effectiveness under various configurations and real-world scenarios.

First, the performance of various initial solution generation methods is presented in Table

Table 5.3 Vehicle specifications

		Setup time (min)		(Un)loading time (min)		Capacity	
Vehicle	Speed (km/h)	Ambulant	Wheelchair	Ambulant	Wheelchair	Seats	Wheelchairs
Sedan	33	1	1.5	1	1.5	2	2
AT	33	1	1.5	1	2	2	1
Friendly	30	1	1.5	1	2	2	2

Table 5.4 Break pattern specifications and maximum route durations

Contract k	Pattern ℓ	n^ℓ	δ^ℓ (min)	$\Delta^{min,\ell}$ (min)	$\Delta^{max,\ell}$ (min)	$T^{max,\ell}$
2	1	2	20	180	360	1080
4	1	2	20	120	360	1080
5	1	2	20	180	300	900
6	1	2	20	180	300	900
7	1	1	20	120	300	600
	2	2	15	120	300	800
	3	2	20	120	300	900
9	1	1	20	120	300	600
	2	2	15	120	300	800
	3	2	20	120	300	900

5.5, where three distinct approaches—greedy-by-route, sliding window, and LP-based heuristics—are compared across several test instances. The results demonstrate that the LP-based method consistently yields superior initial solutions with the lowest costs for all instances, reflecting its effectiveness in providing higher-quality starting points for further optimization. While the LP-based approach requires more computational time than the greedy-by-route method, it outperforms the latter significantly in terms of solution quality. Notably, the LP-based method also exhibits a shorter computational time than the sliding window method for most instances, further highlighting its efficiency. Despite being the fastest, the greedy-by-route method produces the highest initial costs, indicating that its speed comes at the expense of solution quality. The sliding window method offers a middle ground, with costs lower than greedy-by-route but higher than LP-based, while also requiring the longest computational time. Overall, these results suggest that the LP-based method provides the most favorable balance between solution quality and computational effort, particularly for larger problem instances where solution quality is paramount.

It should be noted that to further enhance the initial solution, a simple assignment model was applied to allocate the generated routes to vehicles, accounting for the limitation on the number of available vehicles in each contract. This approach was particularly relevant due to the presence of different cost structures associated with various contracts. The assign-

ment model has been tested on the greedy-by-route and sliding window methods, where it successfully optimized the assignment of routes to vehicles, leading to an improvement of approximately 3% in the initial solution.

To determine an effective sequence of neighborhood classes, we implemented a VNS algorithm that included all the shakers outlined in Section 5.4.2, with varying neighborhood sizes from 1 to 5 for swap and chain neighborhood classes. The performance metrics for each shaker are presented in Table 5.6 using the LP-based heuristic for generating the initial solution. These metrics include the number of calls, the number of improvements made, total improvements, the average improvement per call, the total time spent on each shaker, and the rate of improvement per second. This analysis allows us to rank the shakers based on their efficiency and contribution to solution improvements, aiding in prioritizing the most effective shakers within the VNS framework. It should be noted that swap and chain neighborhoods of sizes 3 to 5 were tested during preliminary experiments, but consistently yielded negligible improvements relative to their computational effort. For this reason, they were excluded from the final neighborhood sequence and are not considered in Table 5.6. The analysis shows that the chain neighborhood shaker with size 1 (C-1) was called the most frequently, with 4,635 calls, yielding a total improvement of 455,546, though its rate of improvement per second (1,511) was not the largest. In contrast, the swap neighborhood shaker with size 1 (S-1) demonstrated a much higher improvement rate per second (9,920), though it was called far fewer times (802 calls). The IP-Based shaker produced the largest average improvement per call (5,012) and the greatest total improvement (1,007,412), but at the cost of significantly more time (653.3 seconds).

The VNS algorithm was configured to solve the problem by applying two termination conditions: (1) a maximum of 200 consecutive iterations without improvement, and (2) the temperature in the acceptance condition reaching a predefined threshold. The first condition ensures that the search process is halted when the solution quality no longer improves after a significant number of repetitions, preventing the algorithm from spending unnecessary time on stagnant areas of the search space. The second condition, related to the acceptance criterion, leverages a temperature-based mechanism often found in simulated annealing-like procedures. As the temperature decreases throughout the algorithm, it becomes increasingly difficult to accept worse solutions, encouraging the algorithm to converge toward an optimal or near-optimal solution. These combined termination conditions help balance the exploration and exploitation phases of the search, ensuring both thoroughness and efficiency in the solution process.

The final performance of the algorithm using the mentioned sequence of shakers, based on

Table 5.5 Performance of initial solution generation methods

	Greedy-by-route		Sliding window		LP-based	
Instance	Initial Cost	Time (min)	Initial Cost	Time (min)	Initial Cost	Time (min)
S1	8,358,104	4.2	8,034,477	10.7	7,808,718	7.9
S2	18,250,140	11.1	17,923,474	28.4	17,205,279	20.6
S3	10,001,434	21.4	9,568,422	54.6	9,051,033	36.1
S4	4,587,502	7.5	4,506,903	18.7	4,304,552	14.0
S5	5,012,846	7.6	4,849,895	19.4	4,635,143	14.2

Table 5.6 Overall performance of the shakers sequence for solving Instance S3

Shaker – Neighborhood size	#calls	#Imp.	Total Imp.	Average Imp.	Time (sec)	Imp. per sec
C-1	4635	182	455,546	2503	301.4	1511
S-1	802	188	409,652	2179	41.3	9920
C-2	1250	167	283,733	1699	96.1	2954
S-2	555	118	193,166	1637	29.8	6480
IP-Based	551	201	1,007,412	5012	653.3	1542
Z	2293	119	291,224	2447	162.6	1791
E	364	21	48,384	2304	29.5	1642

different initial solution methods, is shown in Table 5.7. For each instance, we conducted 15 independent runs of the algorithm to account for its random components. The table reports the best solution value, the average deviation from the best (AvgD), and the worst deviation from the best (WorstD) across the 15 runs, as well as the computing time in minutes. The results show that the proposed algorithm, when generating the initial solution with LP-based method, consistently achieves the lowest final costs across all cases, although this requires more computing time compared to when greedy-by-route method is used. On the other hand, with the greedy-by-route heuristic, the algorithm is slightly faster in most cases but yields higher costs, representing a clear trade-off between speed and solution quality. The sliding window heuristic offers a middle ground, delivering final costs and computing times that fall between the other two approaches. Importantly, the variability across runs is small (the worst deviations remain below 0.5% in all cases) indicating that the algorithm provides stable and robust performance. This comparison underscores the significance of selecting an appropriate initial solution method, depending on the trade-off between computational time and the quality of the final solution. LP-based heuristics are likely preferable when solution quality is the main priority, while the greedy-by-route approach may be more suitable in scenarios where faster results are needed.

Regarding local searches, both the route adjustment and IP-based methods exhibited sim-

Table 5.7 Performance of the proposed algorithm with different initial solution generation methods

	Greedy-by-route				Sliding window				LP-based			
Instance	Best	AvgD (%)	WorseD (%)	Time (min)	Best	AvgD (%)	WorseD (%)	Time (min)	Best	AvgD (%)	WorseD (%)	Time (min)
S1	7,423,796	0.17	0.40	15.2	7,435,731	0.19	0.41	19.8	7,409,878	0.17	0.39	17.9
S2	15,380,049	0.20	0.38	33.5	15,398,348	0.21	0.40	44.1	15,342,872	0.18	0.30	38.9
S3	6,049,034	0.22	0.32	58.3	6,060,114	0.22	0.34	75.9	6,039,101	0.16	0.24	61.9
S4	2,743,667	0.20	0.27	22.4	2,748,187	0.22	0.29	29.0	2,735,246	0.18	0.25	26.5
S5	3,052,998	0.16	0.31	22.8	3,061,038	0.17	0.33	30.2	3,037,664	0.12	0.31	26.9

ilar performance, each contributing to approximately a 10% total improvement in the final solution. Notably, there was no significant difference between the two methods in terms of execution time. This consistency in both solution quality and computational efficiency indicates that either approach could be effective depending on the specific requirements of the problem, with minimal variation in performance.

To evaluate the effect of each shaker family within the presented sequence, we conducted a set of 10 experimental runs on instance S3, each time removing a specific shaker family to observe its impact on the algorithm's performance. By excluding one family of shakers at a time, we aim to understand how each family contributes to the overall solution and how they interact and complement one another within the sequence.

The results presented in Table 5.8 illustrate the critical role that each family of shakers plays in the performance of the proposed algorithm. The values indicate the amount of improvement achieved by each shaker during the algorithm's execution, reflecting their contribution to enhancing the solution quality. The gap shown in Table 5.8 represents the percentage difference between the solution obtained by removing each shaker family and the solution from the full version of the algorithm, where all shakers are included. The total time of the algorithm for each of these scenarios is also presented in the last row of the table. The removal of any family shaker led to a deterioration in the final solution quality, with the extent of the impact varying across the different shakers. Notably, the removal of the IP-based shaker had the most significant effect, resulting in a 12.4% increase in the gap from the full version solution, underscoring its importance in guiding the search process. Similarly, the exclusion of the chain (C) and swap (S) shakers led to substantial increases in the gap, at 7.4% and 6.2% respectively, indicating that these neighborhoods play a vital role in exploring the solution space effectively. The removal of the natural shaker (Z) also had a noticeable impact, with a 4.1% gap, reflecting its contribution to improving the solution. On the other hand, the removal of the exchange-vehicle (E) shaker, while still leading to a gap, resulted in a relatively modest increase of only 0.7%, suggesting that its role, while important, is less

central than the other shakers. The total runtime analysis shows that removing the IP-based shaker had the most significant effect, reducing the total time to 52.9 minutes, but at the cost of obtaining a worse solution, highlighting its critical role in improving solution quality. Similarly, while the removal of other shakers generally led to a reduction in runtime, except for the swap (S) shaker, which increased the total time to 62.7 minutes, these reductions came at the expense of solution quality, emphasizing the importance of retaining all shakers for achieving better results. These findings emphasize the complementary nature of the shakers and demonstrate that the algorithm's full potential is realized when all of these neighborhood structures are incorporated. Each shaker contributes to the overall efficiency of the search, and their combined use ensures the generation of higher-quality solutions.

5.5.3 Result of medium-size instances

The performance of the proposed algorithm on medium-sized problem instances (with the number of requests ranging between 300 and 849, and the number of vehicles varying from 25 to 81) is evaluated in comparison with the exact branch-price-and-cut (B&P&C) algorithm by Karimi et al. [114]. As shown in Table 5.9, the proposed algorithm demonstrates a remarkable balance between solution quality and computational efficiency. The gap between the solutions obtained by the proposed algorithm and the optimal solutions found by the B&P&C method is consistently below 0.5%, with an average gap of only 0.26%. Despite this small gap, the proposed algorithm significantly outperforms the B&P&C method in terms of computational time, with run times reduced by several orders of magnitude. For example, in Instance P11, the proposed algorithm achieved a solution with a 0.14% gap to the optimal value in just 10.7 minutes, whereas the B&P&C method required 486 minutes to find the optimal solution. This substantial reduction in computational time highlights the effectiveness of the proposed heuristic in delivering near-optimal solutions in a fraction of the time.

5.6 Conclusion

In this paper, we introduced a VNS algorithm for solving very large-scale practical instances of the DARP. By integrating advanced initial solution heuristics, including an LP-based method, and enhancing the algorithm with mixed-integer programming techniques, we were able to significantly improve the quality of solutions while maintaining competitive computational times. The proposed algorithm demonstrated its robustness through extensive testing on real-world instances, including very large problems with 10,527 requests. The algorithm consistently produced high-quality solutions in less than an hour for very large-scale

Table 5.8 The impact of removing family shakers on algorithm performance for solving Instance S3

Shaker – Neighborhood size	Full version	Without C	Without S	Without IP-based	Without Z	Without E
C-1	455,546	-	535,133	507,231	458,437	456,781
S-1	409,652	502,301	-	422,495	414,908	410,368
C-2	283,733	-	338,389	317,907	289,534	284,874
S-2	193,166	244,140	-	219,981	200,665	192,140
IP-Based	1,007,412	1,043,276	1,044,137	-	1,027,887	1,002,721
Z	291,224	404,931	344,890	424,468	-	300,343
E	48,384	50,293	50,517	49,721	49,086	-
Gap with full version	-	7.4%	6.2%	12.4%	4.1%	0.7%
Total time (min)	61.9	56.8	62.7	52.9	59.5	61.3

Table 5.9 Comparative results on medium-sized instances

Instance	#Requests	#Vehicles	Proposed VNS solution	Gap with optimal	B&P&C Time (min)	Proposed VNS Time (min)
P1	300	25	84,606	0.00%	134	2.4
P2	330	27	107,516	0.00%	160	2.9
P3	350	29	136,012	0.22%	178	3.9
P4	400	33	150,932	0.00%	237	4.4
P5	500	42	229,948	0.43%	259	5.2
P6	550	48	285,098	0.50%	336	6.4
P7	600	52	372,413	0.47%	318	7.2
P8	700	58	442,992	0.39%	391	8.1
P9	800	62	524,081	0.48%	416	9.0
P10	820	65	639,022	0.22%	435	9.9
P11	849	81	1,743,820	0.14%	486	10.7

instances, demonstrating its effectiveness and efficiency in handling complex real-world scenarios. In addition, the results for medium-sized instances clearly indicate the efficiency of the proposed approach, achieving near-optimal solutions in a fraction of the time required by exact methods such as branch-price-and-cut algorithms. Furthermore, the analysis of various initial solution strategies and shaker sequences revealed important insights into the trade-offs between solution quality and computational time, offering practical guidance for the selection of appropriate strategies in real-world applications. Overall, the proposed VNS algorithm provides a valuable and scalable solution for DARP, making it a suitable tool for large-scale transportation systems. Future work may explore additional enhancements, such as adaptive mechanisms for dynamically adjusting the neighborhood search process or incorporating machine learning techniques to further improve computational efficiency and solution quality in even more complex settings.

Acknowledgements

We thank the personnel of GIRO Inc. for fruitful discussions and for providing real-world datasets. We gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada under the Discovery grants 2023-03791 and 2020-06903.

CHAPTER 6 ARTICLE 3: A VARIABLE NEIGHBORHOOD SEARCH ALGORITHM FOR THE ELECTRIC DIAL-A-RIDE PROBLEM WITH REALISTIC CHARGING CONSTRAINTS

Authors: Mohammad Karimi, Guy Desaulniers, Michel Gendreau

Note: Submitted to Transportation Research Part B: Methodological on June 11, 2025

6.1 Introduction

The dial-a-ride problem (DARP) is a well-known combinatorial optimization problem that involves designing efficient transportation schedules for passengers with specific pickup and drop-off demands [4]. It has widespread applications in paratransit services, elderly and disabled transportation, and ride-sharing systems [5–7]. In recent years, the increasing emphasis on sustainability and environmental concerns has led to the emergence of the electric dial-a-ride problem (E-DARP), where fleets of electric vehicles (EVs) are used instead of conventional fuel-based vehicles to reduce carbon emissions and operational costs [116, 117]. While using EVs contributes to sustainability goals, it introduces additional operational complexities, including limited battery capacity, the need for recharging at specific locations, and non-linear charging behaviors [118]. These factors, along with constraints related to charging station availability and scheduling, introduce additional complexities that significantly increase the difficulty of solving the problem, particularly for large-scale instances. Moreover, balancing service quality with battery management constraints necessitates sophisticated optimization techniques that can efficiently handle large solution spaces. Addressing these challenges is crucial for the practical deployment of electric on-demand transportation services in urban and suburban areas, where demand is high, and operational efficiency is paramount.

In this paper, we address the E-DARP by integrating several novel features to better reflect real-world conditions. Key innovations include a concave piecewise linear charging function that models the diminishing charging rate of electric vehicles as they approach full battery capacity, as well as the consideration of charging station capacity, which has not been explored in the E-DARP literature before. This assumption reflects real-world scenarios where companies either operate dedicated charging stations or reserve limited capacity at selected public stations. These features ensure that vehicles avoid delays due to overcrowding at charging stations while optimizing energy consumption and minimizing charging time. In addition,

this problem accounts for different types of charging infrastructure (e.g., fast and slow chargers), time-dependent charging pricing strategies, and partial charging policies, improving routing efficiency. To solve this problem, we propose a variable neighborhood search (VNS) algorithm that incorporates novel neighborhood structures, adaptive search strategies, and a new model for charging station insertion. The proposed heuristic is tested on benchmark instances involving up to 900 vehicles and 10,000 requests, demonstrating its effectiveness in large-scale, real-world scenarios by balancing operational constraints with passenger service quality.

The remainder of this paper is organized as follows. Section 6.2 reviews the related literature on the E-DARP, and Section 6.3 presents a detailed description of E-DARP. Then, Section 6.4 outlines the proposed algorithm in depth. Section 6.5 introduces the data used and presents our computational results on different aspects of the problem. Finally, Section 6.6 concludes the paper with a summary of our findings.

6.2 Literature review

Research on the E-DARP has advanced significantly in recent years, with studies investigating diverse modeling frameworks, exact and heuristic solution methods, and real-world applications. A key focus has been on extending the classical DARP to account for the unique challenges posed by EVs—particularly battery limitations, energy consumption, and recharging strategies—while still addressing traditional constraints such as vehicle capacity, time windows, and maximum ride durations. Masmoudi et al. [42] introduced E-DARP in the context of non-emergency healthcare transportation, incorporating multiple types of EVs with resource constraints and a battery-swapping technique to simplify recharging operations. They proposed an evolutionary VNS algorithm capable of handling problem instances involving up to 3 vehicles and 18 requests. Bongiovanni et al. [15] extended the problem by introducing the electric autonomous dial-a-ride problem (E-ADARP), incorporating autonomous EVs, multiple depots, and state-of-charge constraints. They formulated the problem using mixed-integer programming models and solved it using a branch-and-cut (B&C) algorithm with problem-specific valid inequalities. The largest instance that their B&C algorithm can solve to optimality includes 5 vehicles and 40 requests. Later, Bongiovanni et al. [95] investigated a dynamic version of E-ADARP and introduced a machine learning-based large neighborhood search heuristic, where machine learning techniques were used to guide the selection of destroy-repair operators, significantly improving computational efficiency. Su et al. [98] presented a deterministic annealing local search approach to solve the E-ADARP that yields near-optimal solutions within practical computational times. They also develop

a larger benchmark instance set and report results for instances involving up to 8 vehicles and 96 requests.

Su et al. [96] proposed a fragment-based path representation combined with a column generation approach, transforming feasible route fragments (i.e., sequences of pickups and deliveries starting and ending with an empty vehicle) into arcs to construct a compressed graph while enforcing partial recharging constraints to minimize excess ride time. They provided new lower bounds for instances with up to 8 vehicles and 96 requests. Limmer [97] introduced a bilevel large neighborhood search in which the outer level optimized charging station insertion while the inner level handled request assignments, demonstrating its effectiveness on large-scale instances with up to 5,200 requests and 260 vehicles. Molenbruch et al. [43] studied a fixed-circuit version of E-DARP, optimizing shuttle schedules and recharging while developing polynomial-time algorithms to solve subproblems efficiently. Bresich et al. [99] introduced a large neighborhood search heuristic leveraging battery-restricted fragments for efficient route representation, achieving new best-known solutions for several benchmark instances. Su et al. [100] proposed a labeling algorithm based on fragment-based abstraction, integrating it into a branch-and-price framework. Stallhofer & Parragh [101] presented a mixed-integer linear programming model for E-ADARP, utilizing an event-based graph to implicitly enforce key constraints and strengthen the model with both existing and new valid inequalities. Recently, Dong et al. [1] explored the E-DARP with time-of-use electricity pricing and proposed a mixed-integer programming model, along with an adaptive large neighborhood search heuristic, to optimize routing and charging strategies. Their model and algorithm were validated with computational experiments on benchmark and real-world cases with up to 10,000 requests and 900 vehicles.

Despite significant progress in the literature, existing studies often assume homogeneous battery capacities with fully charged vehicles at the start of planning and typically consider only a single type of charging infrastructure. Moreover, most works apply linear charging functions and overlook operational constraints such as charging station capacities. To bridge these gaps, our study considers a more realistic and comprehensive problem that incorporates multiple types of charging infrastructure (e.g., fast and slow chargers), time-dependent charging pricing, and a partial charging policy. Most notably, we introduce a concave piecewise linear charging function to capture realistic charging dynamics better, and we are also the first to explicitly model charging station capacity constraints within the E-DARP. These contributions enhance the model's applicability to real-world electric mobility systems. To solve this problem, we propose a VNS algorithm tailored for large-scale E-DARP instances. The algorithm integrates novel neighborhood structures, adaptive search strategies, and a new model for charging station insertion, alongside efficient heuristics designed to improve

both solution quality and computational performance. By incorporating more realistic operational constraints, such as station capacity and non-linear charging behavior, the proposed approach offers a robust and scalable solution framework for real-world electric dial-a-ride systems.

6.3 Problem definition

The E-DARP extends the classical DARP by integrating the operational constraints specific to EVs. In this setting, a set of transportation requests—each defined by a pickup and delivery location, associated time windows, and service requirements—must be fulfilled by a fleet of capacitated EVs. The objective is to design cost-efficient routes that service all requests while satisfying operational constraints related to vehicle capacities, maximum user ride times, and time window compliance. Unlike conventional DARP formulations, the E-DARP introduces additional complexity by incorporating battery constraints. Each vehicle operates with a finite battery capacity, and energy consumption is typically modeled as a function of traveled distance. To ensure route feasibility, recharging operations at designated charging stations may be required, which introduces further temporal and routing considerations.

The E-DARP is defined on a complete directed graph $G = (V, A)$, where V represents the set of vertices and A is the set of arcs. The vertex set V is partitioned into five subsets: $V = P \cup D \cup S \cup O \cup F$, where $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ denote the sets of pickup and drop-off nodes, respectively, associated with a set of requests $N = \{(i, i+n) \mid i \in P\}$. The set S represents charging stations, while O and F are the sets of origin and destination depots, respectively. Each request $i \in N$ is associated with a maximum ride time L_i. A time window $[e_i, l_i]$ is specified for each node $i \in P \cup D \cup O \cup F$, where e_i and l_i denote the earliest and latest possible start times for service at node i. Each node $i \in V$ also has a service duration s_i and an associated load q_i, where $q_i > 0$ for pickup nodes $i \in P$, $q_{i+n} = -q_i$ for the corresponding drop-off nodes $i + n \in D$, and $q_i = 0$ for all other nodes $i \in S \cup O \cup F$.

Travel times $t_{i,j}$ are defined for all arcs $(i, j) \in A$, and battery consumption $\beta_{i,j}$ is determined using an energy consumption model based on travel time. The fleet of EVs is homogeneous in terms of capacity Q and battery capacity C (expressed in kWh) for each vehicle $k \in K$, and may have varying initial battery levels B^k. Due to their limited battery capacities, EVs may need to recharge at charging stations during their routes. Charging operations can occur at any charging station $i \in S$, where partial recharging is allowed (i.e., recharge can stop at any time, not only when the battery is fully charged). Each charging station features:

- **Charging function**: A concave piecewise linear charging function $\mathcal{F}_i$ with a set $\mathcal{P}$

of segments, where $\mathcal{P} = \{1, \ldots, p\}$, which maps the state-of-charge (SoC) level upon arrival b_i and the charging duration Δ_i to the SoC level upon departure. In this function, each piece $p \in \mathcal{P}$ begins at a battery level ϕ_{p-1} and ends at ϕ_p, where $0 = \phi_0 < \phi_1 < \cdots < \phi_p = C$ and the recharging rate θ_p (energy charged per timestep) for each piece $p \in \mathcal{P}$ is given by $\theta_1 > \theta_2 > \cdots > \theta_p > 0$ (Figure 6.1).

- **Charger availability**: The number of available chargers R_i at each station i limits the maximum number of vehicles that can be charging simultaneously during any time period $t \in \mathcal{T}$.
- **Time-dependent charging costs**: Charging costs vary by time period t, denoted by $c_{i,t}$, reflecting dynamic electricity prices throughout the day.

A feasible solution to the E-DARP must satisfy several critical constraints. First, each customer must be served exactly once by a single vehicle, and this service must adhere to vehicle load limitations, customer-specific time windows, and maximum ride times. Second, every vehicle route must be energy-feasible, meaning that the battery level of an EV must remain within the allowable range $[0, C]$ throughout the entire route. Third, the charging infrastructure must be respected: at any given period t, the number of vehicles charging at a particular station i cannot exceed the station's capacity R_i. In addition, the battery level of each vehicle upon arrival at its destination depot must be greater than or equal to a predefined minimum SoC. The objective function is defined as a weighted sum of the total travel time of all vehicles and the excess ride time of users, allowing direct quantification of user inconvenience. Including excess ride time in the objective can help improve service quality by reducing delays without incurring additional operational costs.

6.4 The proposed algorithm

The E-DARP is tackled through a VNS heuristic designed to explore the solution space effectively and efficiently. The procedure initiates with a feasible solution, denoted as s_{init}, which serves as both the incumbent and current solution s. The algorithm then systematically explores a predefined sequence of neighborhood classes, $N_k(s)$, to generate a neighboring solution s'. If s' satisfies the conditions for further exploration (as detailed in Section 6.4.3), a local search procedure is applied to refine it, yielding an enhanced solution s''. Should s'' outperform the current incumbent, it replaces s as the new best-known solution. Otherwise, the incumbent remains unchanged, and an acceptance mechanism is employed to determine whether to continue from s''. This iterative process continues until a predefined stopping

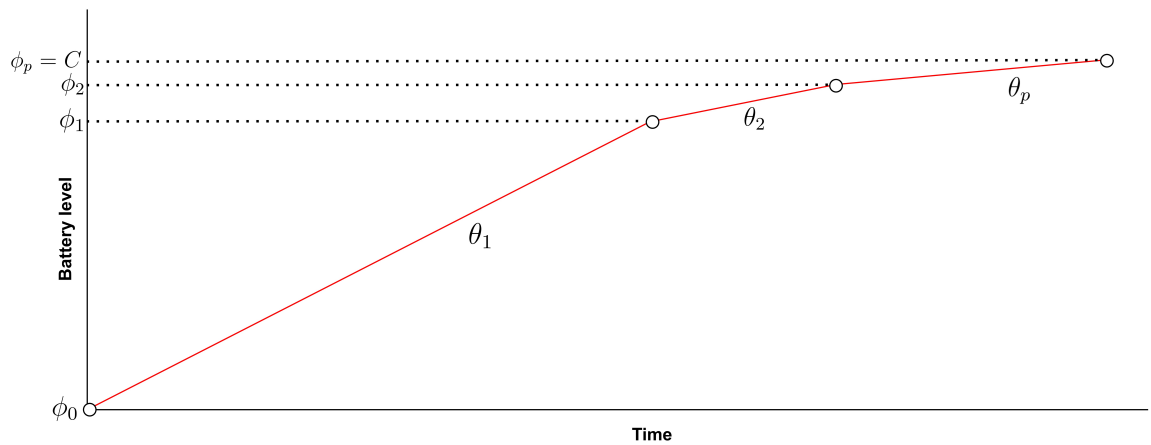


Figure 6.1 The concave piecewise linear charging functions

criterion is met, ensuring the algorithm explores diverse regions of the solution space. The overall framework of the proposed VNS heuristic is outlined in Algorithm 3, which embodies both diversification through shaking and intensification via local improvement.

The following subsections describe the methods used in our VNS algorithm. Section 6.4.1 explains the approach for generating the initial solution. Section 6.4.2 outlines the neighborhood searches employed to explore the solution space. Finally, Section 6.4.3 presents the local search technique to refine the generated solutions.

6.4.1 Generating the initial solution

In the literature, various methods have been proposed for generating initial solutions, recognizing that a high-quality starting point is critical for the efficiency and effectiveness of metaheuristic frameworks. To this end, we propose an LP-based heuristic for constructing an initial feasible solution, as outlined in Algorithm 4. The method discretizes the planning horizon into hourly intervals, allowing the problem to be addressed incrementally over manageable sub-periods. At the beginning of each period, starting from the earliest, a DARP model inspired by the arc-flow formulation of Cordeau [61] is constructed using the set of available vehicles and active requests (requests whose pickup node time window intersects the period), ensuring that only relevant requests are considered for service within that time frame. This model includes binary decision variables x_{ij} indicating the presence of vehicle flow along arc (i, j). To reduce computational complexity, the linear relaxation of the model is first solved. The resulting fractional values $\bar{x}_{ij}$ are then used to guide a variable cost adjustment using the transformation $c''_{ij} = \frac{c'_{ij}}{\bar{x}_{ij}}$, where c'_{ij} and c''_{ij} are the original and adjusted costs, respectively. The adjusted model is re-solved iteratively over a fixed number

Algorithm 3 VNS heuristic

```
// initial solution
Generate s_init;
Set s := s_init; Set k := 1; Set s_best := s;
repeat
    // shaking
    Compute s' in neighborhood N_k(s);
    // local search
    if s' is a promising solution then
        Apply local search to s' yielding s'';
    else
        Set s'' := s';
    end if
    if s'' is better than s then
        Set s := s''; Set k := 1;
        if s'' better than s_best then
            Set s_best := s'';
        end if
    else
        Set k := k + 1;
        if acceptance criteria are satisfied then
            Set s := s'';
        end if
    end if
until some stopping criterion is met
return s_best;
```

of iterations, denoted by $MaxIteration$, allowing for progressive refinement in arc selection. Binary variables that consistently remain inactive (i.e., with zero values) across these iterations are fixed to zero, effectively reducing the model's size. The resulting reduced model is then solved as an integer program within a specified time limit, yielding an optimized sub-solution for the given time period. This process is repeated sequentially for all subsequent periods, with each iteration updating the state of the fleet based on previous assignments. This time-segmented approach enables the generation of a computationally tractable and practically effective initial solution that balances problem scale and solution quality.

6.4.2 Neighborhood classes

The proposed algorithm employs four distinct neighborhood classes (or shakers), each designed to explore different regions of the solution space. These neighborhood structures are tailored to the specific characteristics of the E-DARP, and one of them incorporates an

Algorithm 4 LP-based heuristic for initial solution generation

```
Divide the length of the day into periods;
Set per := 1;
repeat
    Set iteration := 1;
    Set c'_ij := c_ij;
    Set c''_ij := c_ij;
    Formulate a DARP model for active requests and vehicles in period per
    Set ActiveVariables := ∅;
    repeat
        Solve linear relaxation of the model with cost coefficients c'_ij;
        Set ActiveVariables := ActiveVariables ∪ {(i, j) : x̄_ij > 0};
        Set c''_ij := c'_ij / x̄_ij for (i, j) such that x̄_ij > 0};
        Set c'_ij := c''_ij;
        Set iteration := iteration + 1;
    until iteration ≤ MaxIteration
    Fix inactive variables ((i, j) ∉ ActiveVariables) to zero;
    Solve the restricted integer program in a limited time;
    Set per := per + 1;
until per ≤ TotalPeriods
return initial solution;
```

integer programming (IP)-based neighborhood search method to refine promising solutions further. Most of the shaker methods are parameterized by a size parameter, which governs the maximum number of requests or routes that may be altered. These neighborhood classes are described as follows.

Swap Neighborhood (S): The Swap shaker, initially introduced by Parragh et al. [31], facilitates the exchange of requests between two distinct routes. The procedure begins by selecting two different routes using a roulette wheel selection strategy (as described in Section 6.4.2), which ensures a probabilistically diverse sampling of candidate routes. For each selected route, a subsequence of requests is randomly identified, determined by both the starting position and a random length constrained by a predefined maximum size parameter. To preserve route feasibility, the entire request—including both origin and destination vertices—must be included in the swap, even if part of it lies outside the selected subsequence. Once the sequences are identified, all requests within the respective segments are removed from their current routes. Subsequently, these requests are reinserted into the opposing route, one by one, with feasibility checks enforced throughout the insertion process.

Chain Neighborhood (C): The Chain shaker, also introduced by Parragh et al. [31], is inspired by the ejection chain mechanism and facilitates a cascading reassignment of request sequences

across multiple routes. The process begins with the random selection of a sequence of vertices from a given route, similar to the approach used in the Swap neighborhood. This sequence is then transferred to a second route, which is probabilistically chosen using the roulette wheel selection method outlined in Section 6.4.2, thereby encouraging diversification in route selection. A random sequence length l is drawn, constrained by the maximum shaker size, which defines the length of the request sequence to be moved. Subsequently, a sequence of length l is selected from the second route—again using the roulette wheel—and transferred to a third, randomly chosen route. This process continues iteratively, with each new route receiving a sequence from the previous one, until the maximum number of allowed sequence transfers (defined by the shaker size) is reached. The shaker size thereby governs both the number of sequence movements and their respective lengths, facilitating a rich and structured exploration of the solution space.

IP-based Neighborhood: This shaker employs an optimization approach within a limited subset of the solution space. It begins by selecting two distinct routes using the roulette wheel selection mechanism. Based on the division of the planning horizon into hourly intervals, the algorithm identifies the time period with the highest number of shared requests between the two selected routes. To ensure meaningful re-optimization, each route must contain at least one request within the chosen time window. Once the period is determined, a DARP subproblem is formulated over the two routes, restricting flexibility to only those requests whose pickup times fall within the selected period. All other parts of the routes are held fixed. Importantly, if a delivery occurs in the current period but the corresponding pickup took place in a previous period, the request is considered fixed. Conversely, requests picked up during the selected period—even if their delivery occurs in a subsequent period—remain part of the model. The resulting subproblem is modeled as a mixed-integer program, and a MIP solver is applied to determine the optimal reassignment of requests between the two routes for the selected period. This shaker thus enables highly targeted improvements by leveraging the precision of exact optimization while containing computational complexity through time- and route-based restriction of the decision space.

All Natural Sequences Combinations Neighborhood (Z): This neighborhood class builds on the concept of *natural sequences*, originally introduced within the context of the zero-split neighborhood by Parragh et al. [31]. A natural sequence is defined as a contiguous set of vertices along a route for which the cumulative vehicle load returns to zero at the end of the sequence, thus forming a self-contained segment of service. In this shaker, proposed by Muelas et al. [79], each natural sequence is treated as a single unit, and all possible pairwise swaps of natural sequences between different routes are systematically evaluated. For instance, if two routes, i and j, each contain two natural sequences, the algorithm examines all four

possible swap combinations between these sequences. This exhaustive evaluation ensures a thorough exploration of the swap space between route pairs. Unlike other neighborhood classes, this shaker does not rely on a size parameter, as it consistently evaluates the complete set of feasible natural sequence swaps. Its primary objective is to enhance solution quality by leveraging structurally significant subcomponents of routes, thereby facilitating meaningful reassignments across the fleet.

The described neighborhood classes can be applied in varying sequences and with differing neighborhood sizes, thereby offering considerable flexibility in the algorithm's implementation. Nevertheless, in accordance with the foundational principles of the VNS framework, it is recommended to begin the search process with shakers that induce minimal perturbations to the incumbent solution. This strategy is rooted in the core philosophy of VNS, which advocates for a systematic exploration that initially emphasizes small, incremental modifications to preserve promising structural attributes of the current solution. As the search progresses, increasingly disruptive neighborhoods are introduced to escape local optima and promote broader diversification. Adopting such a hierarchical application of neighborhood classes ensures a balanced trade-off between intensification and diversification, thereby enhancing the overall search efficiency and the likelihood of attaining high-quality solutions with reduced computational expense.

Roulette wheel route selection

To effectively guide the application of neighborhood classes, we employ a roulette wheel selection strategy in conjunction with the concept of route stability introduced by Parragh et al. [13]. In this framework, each route is associated with a stability counter that records the number of consecutive iterations during which the route has remained unmodified. Routes with higher stability values are considered more likely to be trapped in local optima and are thus prioritized for perturbation. The selection mechanism is probabilistic: the stability counters are summed to obtain a total stability score, and each route's probability of being selected is proportional to its individual stability counter relative to this total. Consequently, routes that have not undergone modifications for extended periods are more likely to be chosen for neighborhood perturbation, while ensuring that all routes retain a non-zero probability of selection, thereby preserving the diversity of the search. The roulette wheel procedure operates by generating a random number between zero and the total stability score. The algorithm sequentially accumulates the stability counters of the routes until the cumulative value surpasses the generated random number, at which point the corresponding route is selected. Upon application of a shaker to the selected route, its stability counter is

reset to zero, while the stability counters of all other routes are incremented. This adaptive mechanism dynamically balances the focus on potentially stagnant routes with exploratory diversification, enhancing the overall effectiveness of the search in escaping local optima and improving solution quality.

6.4.3 Local search

In contrast to previous neighborhood classes that primarily focus on inter-route modifications, local search methods concentrate on intra-route optimization by greedily enhancing the sequence of vertices within individual routes. Given the computationally expensive nature of local search, it is not applied after every shaking step. Rather, it is reserved for *promising* solutions—those that exhibit potential for generating new incumbent solutions. Based on empirical results, a solution (s') is deemed *promising* if its cost does not exceed the current solution (s) by more than 1%. Various local search algorithms have been developed for the DARP, each employing distinct strategies for route optimization. In this work, we propose an IP-based method to refine routes by optimizing timing, reducing travel distance, and maximizing vehicle capacity utilization, with the goal of enhancing overall solution quality efficiently. The local search algorithm constructs and solves the DARP model for a single route, focusing on requests within a specific time interval. Extensive experimentation reveals that the most effective approach for selecting the time interval is to choose the one-hour period with the highest request count, along with its two adjacent periods. This method ensures the inclusion of a broader set of requests, facilitating better optimization across neighboring time slots and thereby improving overall routing efficiency.

6.4.4 Charging station insertion

To ensure the feasibility of electric vehicle routes with respect to energy constraints, a dedicated charging station insertion procedure is proposed, shown in Algorithm 5. This method begins with a feasible route that satisfies all operational constraints but excludes charging considerations. The primary objective is to evaluate battery consumption along the route, identify points where the energy level becomes critical and when there is no onboard request, and determine optimal locations and durations for charging. The process involves simulating SoC throughout the route, locating feasible charging stations within proximity to critical points, and solving a linear relaxation model to guide the insertion of charging stops. The resulting route is then updated to reflect the added charging activities, ensuring both energy feasibility and service constraints are maintained. For potential charging stations, the time window is determined according to the dispatch time of neighboring nodes, such that the

lower limit is equal to the dispatch time of the previous node (D_i) plus the service time at this node (s_i) and the travel time between this node and the charging station ($t_{i,v}$). The upper limit is equal to the dispatch time of the next node (D_{i+1}) minus the travel time between this node and the charging station ($t_{v,i+1}$), shown in Figure 6.2. It should be noted that if this time window is not feasible, this location will be eliminated for potential charging. This charging station insertion procedure is applied to each route immediately after it is constructed or improved during the solution process, to ensure energy feasibility before any cost evaluation or solution acceptance.

To determine the optimal amount of energy to be charged at candidate stations, two versions of the insertion model are considered based on the characteristics of the charging process. In the first version, a constant charging rate is assumed, resulting in a linear relationship between charging time and energy gained. This simplification allows for a straightforward linear programming formulation. In contrast, the second version incorporates a concave piecewise linear charging function, which more accurately reflects real-world charging behavior where the rate of energy transfer decreases as the battery approaches full capacity. This version introduces additional complexity to the model but enables more realistic and efficient charging schedules. Both formulations are solved as linear relaxations to support the efficient selection of charging stations under time and energy constraints. A detailed explanation of each insertion model is provided in the remainder of this section.

Prior studies on the routing problem have proposed dynamic programming (DP) procedures for the insertion of charging stations, particularly under concave piecewise linear charging functions, which optimally determine both the location and amount of energy charged, considering diminishing returns of longer charging durations. Notable examples include the work of Kullman et al. [119] and Dong et al. [1], where DP exploits the problem's sequential decision structure and concavity to efficiently explore feasible SoC transitions. In contrast, the method proposed in this study is an approximate insertion approach that solves a linear optimization model to determine feasible charging locations and durations.

Insertion model with constant charging rate

Considering the set ∇ of potential charging stations along the route, which are indexed from 1 to $|\nabla|$, the insertion model under a linear (constant-rate) charging function α is formulated as follows:

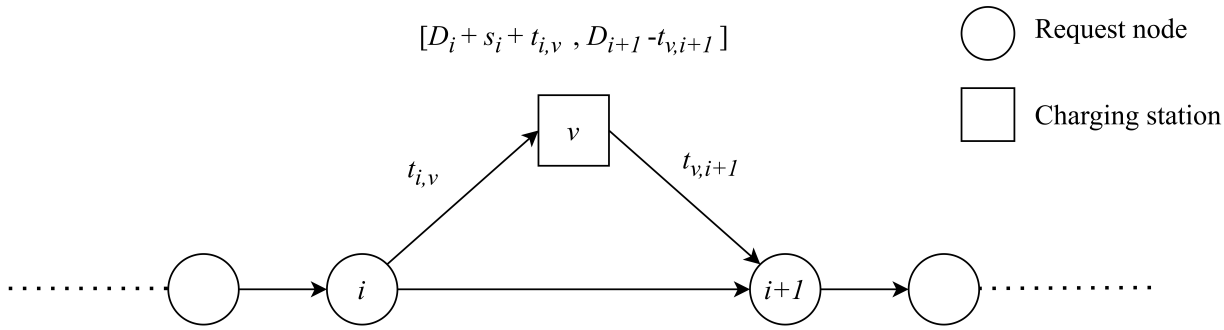


Figure 6.2 The nearest charging station time window

Algorithm 5 Charging station insertion method

```
Set r* := r
Compute energy consumption over r
Identify potential insertion points for charging
for each insertion point do
    Identify nearby feasible charging stations
    Compute available charging time windows based on travel and service constraints
end for
Solve insertion model over candidate stations
if model is feasible then
    Select stations with positive charging times
    Insert selected charging stations into route r*
    Update route timing, time windows, and battery levels
    return updated route r*
else
    return route is infeasible
end if
```

$$\min \quad \sum_{v \in \nabla} T_v \tag{6.1}$$

$$\text{s.t.} \quad B_v^{\text{new}} = B_v + \alpha T_v, \qquad \forall v \in \nabla \tag{6.2}$$

$$B_{v+1} = B_v^{\text{new}} - \beta_{v,v+1}, \qquad \forall v \in \nabla \tag{6.3}$$

$$B_v^{\text{new}} \leq C, \qquad \forall v \in \nabla \tag{6.4}$$

$$B_v^{\text{new}} \geq \beta_{v,v+1} + \varepsilon, \qquad \forall v \in \nabla \tag{6.5}$$

$$0 \leq T_v \leq L_v - E_v \qquad \forall v \in \nabla \tag{6.6}$$

$$B_1 = C - \beta_{\text{depot},1}. \tag{6.7}$$

where:

- T_v: time spent charging at station v,
- B_v: battery level before charging at station v,
- B_v^{new}: battery level after charging at station v,
- $\beta_{v,v+1}$: energy required to reach the next station,
- α: charging rate (energy per unit time),
- C: vehicle battery capacity,
- ε: positive value to ensure feasibility (or minimum battery level at destination depot),
- E_v, L_v: earliest and latest time bounds for charging at station v.

This model minimizes the total time spent at charging stations (6.1). Equation (6.2) calculates the post-charging battery level based on the available charging, charging rate, and the stop duration at the charging station. Constraint (6.3) determines the battery level at each station based on the previous station's battery level and charge level, and the battery consumption between these two stations. Constraint (6.4) limits the battery level at each station to the vehicle battery capacity, and constraint (6.5) ensures that the battery level at each station is sufficient to reach the next charging station (or the end of the route). Constraint (6.6) determines the amount of time that can be spent charging at each station, given the time window of that station. Constraint (6.7) initializes the battery level at the first visited node by subtracting the energy consumption required to travel from the depot to the first potential charging station, ensuring that the route's energy feasibility is accurately modeled. It should be noted that the current energy consumption model overestimates the battery usage when a charging station $v \in \nabla$ is considered as a potential insertion point, because the detour energy to and from the station is included in $\beta_{v,v+1}$ regardless of whether the station is actually visited.

Insertion model with concave piecewise linear charging function

Similar to the previous model, assuming the set ∇ represents the potential charging stations along the route, the insertion model based on the concave piecewise linear charging function is formulated as follows.

$$\min \quad \sum_{v \in \nabla} T_v \tag{6.8}$$

$$\text{s.t.} \quad t_v = \sum_{\rho \in \mathcal{P}} \frac{\max\left(\min(B_v, \phi_\rho) - \phi_{\rho-1}, 0\right)}{\theta_\rho} \qquad \forall v \in \nabla \tag{6.9}$$

$$T_v^{\text{new}} = t_v + T_v \qquad \forall v \in \nabla \tag{6.10}$$

$$B_v^{\text{new}} = \sum_{\rho \in \mathcal{P}} \max\left(\min(T_v^{\text{new}}, \psi_\rho) - \psi_{\rho-1}, 0\right) \theta_\rho \qquad \forall v \in \nabla \tag{6.11}$$

$$B_{v+1} = B_v^{\text{new}} - \beta_{v,v+1} \qquad \forall v \in \nabla \tag{6.12}$$

$$B_v^{\text{new}} \leq C \qquad \forall v \in \nabla \tag{6.13}$$

$$B_v^{\text{new}} \geq \beta_{v,v+1} + \varepsilon \qquad \forall v \in \nabla \tag{6.14}$$

$$0 \leq T_v \leq L_v - E_v \qquad \forall v \in \nabla \tag{6.15}$$

$$B_1 = C - \beta_{\text{depot},1} \tag{6.16}$$

where t_v is the time corresponding to the battery level before charging at point v, and T_v^{new} is the time after considering charging time. In addition, ψ_ρ denotes the time breakpoints according to the piecewise linear charging function.

Equation (6.9) determines the time corresponding to the battery level before charging based on each piece of the charging function. Equation (6.10) updates the time level by adding the time spent at the charging station to the time level before charging. Equation (6.11) calculates the new battery level based on charging time. Other constraints are similar to the previous model. The components of the model are shown in Figure 6.3.

To handle the proposed model's nonlinear nature in Equations (6.9) and (6.11), which involves a concave charging function, we reformulate the model using the following constraints.

Equation (6.9) linearization:

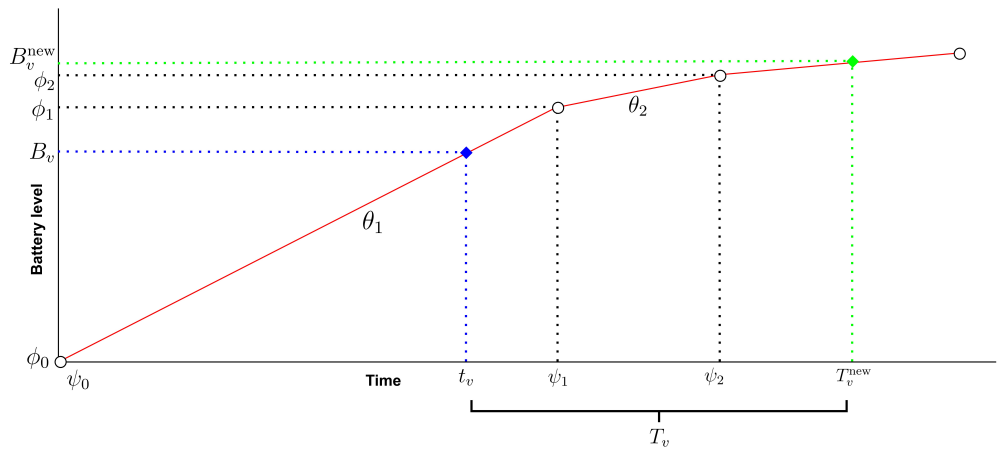


Figure 6.3 The components of the insertion model with concave piecewise linear charging functions

$$z_{\rho v} \leq B_v \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.17}$$

$$z_{\rho v} \leq \phi_\rho \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.18}$$

$$z_{\rho v} \geq B_v - M(1 - \delta_{\rho v}) \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.19}$$

$$z_{\rho v} \geq \varphi_\rho - M\delta_{\rho v} \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.20}$$

$$w_{\rho v} \geq z_{\rho v} - \phi_{\rho-1} \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.21}$$

$$w_{\rho v} \geq 0 \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.22}$$

$$t_v = \sum_{\rho \in \mathcal{P}} \frac{w_{\rho v}}{\theta_\rho} \qquad \forall v \in \nabla \tag{6.23}$$

Equation (6.11) linearization:

$$a_{\rho v} \leq T_v^{\text{new}} \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.24}$$

$$a_{\rho v} \leq \psi_\rho \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.25}$$

$$a_{\rho v} \geq T_v^{\text{new}} - M(1 - \xi_{\rho v}) \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.26}$$

$$a_{\rho v} \geq \psi_\rho - M\xi_{\rho v} \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.27}$$

$$r_{\rho v} \geq a_{\rho v} - \psi_{\rho-1} \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.28}$$

$$r_{\rho v} \geq 0 \qquad \forall \rho \in \mathcal{P}, \forall v \in \nabla \tag{6.29}$$

$$B_v^{\text{new}} = \sum_{\rho \in \mathcal{P}} r_{\rho v} \theta_\rho \qquad \forall v \in \nabla \tag{6.30}$$

where $z_{\rho v}, w_{\rho v}, a_{\rho v}, r_{\rho v} \geq 0$ and $\delta_{\rho v}, \xi_{\rho v} \in \{0, 1\}$. Constraints (6.17)–(6.23) linearize max $(\min(B_v, \phi_\rho) - \phi_{\rho-1}, 0)$ using auxiliary variables $z_{\rho v}$ and $w_{\rho v}$, while constraints (6.24)–(6.30) similarly linearize $\max(\min(T_v^{\text{new}}, \psi_\rho) - \psi_{\rho-1}, 0)$. Binary variables $\delta_{\rho v}$ and $\xi_{\rho v}$ ensure correct selection in the 'min' function, and M is a sufficiently large constant to activate or deactivate constraints conditionally. This linear reformulation allows the insertion model to handle concave charging behavior using mixed-integer programming. It maintains the realism of diminishing charging efficiency while remaining solvable with standard mixed integer programming solvers.

6.4.5 Acceptance criterion

In order to determine whether the incumbent solution should become the new solution (s''), the algorithm employs an acceptance criterion inspired by simulated annealing [79]. Specifically, a solution with a worse objective value is accepted with a probability given by $\exp\left(\frac{-\Delta E}{T}\right)$, where ΔE denotes the difference in the objective function values between the current and new solutions, and T represents the temperature parameter. The temperature parameter T is gradually decreased over time according to a linear schedule, which is determined by the initial temperature and the specified cooling rate. This mechanism facilitates exploration of the solution space while progressively reducing the likelihood of accepting inferior solutions.

6.5 Computational results

The proposed algorithm was coded in Java and uses version 12.4 of the mixed-integer programming solver CPLEX. The computational experiments were performed on a laptop with 8 CPU cores (clocked at 3.2 GHz) and 16 GB of RAM. To evaluate the performance of the proposed algorithm and analyze the influence of various problem features, five different cases are considered. These cases are designed to isolate and assess the impact of the charging process, objective function, and infrastructure capacity. The base case assumes a linear charging process and aims to minimize both travel and ride times. Subsequent cases incorporate more realistic and complex elements such as concave piecewise linear charging functions, different types of charging infrastructure (e.g., fast and slow chargers), station capacity constraints, and the inclusion of time-dependent cost objectives. A summary of the characteristics of instances is provided in Subsection 6.5.1, followed by dedicated subsections that present the detailed results and discussion for each scenario.

6.5.1 Instance description

To assess the performance of the proposed VNS heuristic, extensive computational experiments are conducted on multiple sets of benchmark instances. The first three sets are based on instances used in Limmer [97]. The first set consists of extended versions of 14 problem instances originally introduced by Cordeau & Laporte [3], featuring up to 5 vehicles and 50 requests. Each vehicle can carry up to three passengers, with each request corresponding to a single passenger. The maximum allowable ride time is set to 30 minutes, and the service duration at both pick-up and drop-off locations is fixed at 3 minutes. The second set includes five large-scale instances with up to 260 vehicles and 5200 requests, generated using the same principles as the previous sets. In the second set, energy-related parameters are incorporated to simulate EV operations. Vehicles consume and recharge 0.055 kWh of energy per minute of travel and charging, respectively, with a maximum battery capacity of 14.85 kWh. The objective function combines travel time and excess ride time with corresponding weights of $w_1 = 0.75$ and $w_2 = 0.25$. Additionally, three levels of minimum final battery SoC are considered: $\gamma \in \{0.1, 0.4, 0.7\}$.

To further validate the scalability and effectiveness of the proposed approach, a third set of large and very large instances introduced by Dong et al. [1] is also tested. This set includes 12 instances with problem sizes ranging from 100 vehicles and 1000 requests to 900 vehicles and 10000 requests. In these instances, each request may include 1 to 4 passengers, with a fixed 1-minute service time at pick-up and drop-off points. The vehicles have a charging rate of 0.11 kWh per minute and a discharging rate of 0.055 kWh per kilometer, with a maximum battery capacity of 11 kWh and a passenger capacity limit of 4. The minimum final battery SoC required is set to 0.4 ($\gamma = 0.4$). The instances follow the naming convention `letter-number of vehicles-number of requests`, where a lowercase letter is used for labeling the instance, followed by the number of vehicles and the number of transportation requests.

6.5.2 Base Case - linear charging function

In the first case, the charging process is modeled using a linear charging rate, and the objective is to minimize the total travel time and excess user ride time. This setting is consistent with several previous studies in the literature and serves as a baseline for comparison. By adopting this simplified yet widely used formulation, we aim to evaluate the performance of our proposed VNS heuristic under conditions that closely align with established benchmarks. The results obtained for this base case enable a direct comparison with existing methods and provide insight into the algorithm's effectiveness in scenarios without additional complexities,

such as nonlinear charging behavior or infrastructure constraints. The performance of the proposed algorithm is compared with state-of-the-art methods, including the branch-and-cut algorithm by Bongiovanni et al. [15], the deterministic annealing heuristic by Su et al. [98], and the bilevel large neighborhood search by Limmer [97], shown in Tables 6.1 to 6.4. The bold objective function values are the best solution obtained in this comparison. It should be noted that the presented results correspond to the best solution (i.e., minimum cost) obtained across five independent runs, conducted to account for the random components of the proposed algorithm.

Table 6.1 Results on Cordeau instances with $\gamma = 0.1$

Ins.	Bongiovanni et al. (2019)		Su et al. (2023)		Limmer (2023)		Proposed VNS	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
a2-16	**237.38**	0.02	**237.38**	0.65	238.20	5.00	**237.38**	0.80
a2-20	**279.08**	0.07	**279.08**	1.23	281.00	5.00	**279.08**	0.99
a2-24	**346.21**	0.15	**346.21**	2.68	**346.21**	5.00	**346.21**	1.21
a3-18	**236.82**	0.08	**236.82**	0.42	238.73	5.00	**236.82**	1.07
a3-24	**274.80**	0.23	**274.80**	0.97	275.18	5.00	**274.80**	1.30
a3-30	**413.27**	1.70	**413.27**	0.90	414.88	5.00	**413.27**	1.43
a3-36	**481.17**	1.78	**481.17**	2.54	483.86	5.00	**481.17**	2.69
a4-16	**222.49**	0.06	**222.49**	0.32	**222.49**	5.00	**222.49**	0.69
a4-24	**310.84**	0.52	**310.84**	0.49	311.48	5.00	**310.84**	1.51
a4-32	**393.96**	10.20	**393.96**	0.87	394.66	5.00	**393.96**	1.67
a4-40	**453.84**	8.62	**453.84**	1.53	456.93	5.00	**453.84**	2.22
a4-48	**554.54**	120.00	555.93	2.36	557.24	5.00	**554.54**	3.08
a5-40	**414.51**	19.03	414.80	1.08	415.62	5.00	**414.51**	3.23
a5-50	**559.17**	120.00	561.41	2.29	560.07	5.00	**559.17**	4.21

Table 6.2 Results on Cordeau instances with $\gamma = 0.4$

Ins.	Bongiovanni et al. (2019)		Su et al. (2023)		Limmer (2023)		Proposed VNS	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
a2-16	**237.38**	0.03	**237.38**	0.88	238.20	5.00	**237.38**	0.85
a2-20	**280.70**	0.83	**280.70**	2.35	282.90	5.00	**280.70**	1.04
a2-24	348.04	0.42	**347.04**	3.85	**347.04**	5.00	348.04	1.33
a3-18	**236.82**	0.07	**236.82**	0.44	238.73	5.00	**236.82**	1.12
a3-24	**274.80**	0.28	**274.80**	1.13	275.58	5.00	**274.80**	1.43
a3-30	**413.37**	1.65	**413.34**	1.48	415.51	5.00	**413.37**	1.63
a3-36	484.14	5.11	**483.06**	2.63	485.98	5.00	**483.06**	2.77
a4-16	**222.49**	0.09	**222.49**	0.32	**222.49**	5.00	**222.49**	0.73
a4-24	**311.03**	0.66	**311.03**	0.53	311.48	5.00	**311.03**	1.57
a4-32	**394.26**	11.36	**394.26**	1.05	394.96	5.00	**394.26**	1.72
a4-40	**453.84**	6.96	**453.84**	1.94	457.01	5.00	**453.84**	2.26
a4-48	**554.60**	120.00	558.11	2.96	557.56	5.00	**554.60**	3.19
a5-40	**414.51**	20.35	416.25	1.21	415.63	5.00	**414.51**	3.46
a5-50	560.50	120.00	567.54	2.71	560.41	5.00	**559.51**	4.61

Table 6.3 Results on Cordeau instances with $\gamma = 0.7$

Ins.	Bongiovanni et al. (2019)		Su et al. (2023)		Limmer (2023)		Proposed VNS	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
a2-16	**240.66**	0.09	**240.66**	1.60	242.83	5.00	**240.66**	0.91
a2-20	NA	120.00	**293.27**	2.88	NA	5.00	**293.27**	1.05
a2-24	358.21	16.02	**353.18**	3.44	356.99	5.00	**353.18**	1.39
a3-18	**240.58**	0.80	**240.58**	0.97	242.49	5.00	**240.58**	1.19
a3-24	277.72	2.54	**275.97**	2.06	277.52	5.00	**275.97**	1.59
a3-30	NA	120.00	**424.93**	1.30	432.27	5.00	**424.93**	1.76
a3-36	**494.04**	120.00	**494.04**	2.09	496.75	5.00	**494.04**	3.01
a4-16	**223.13**	1.12	**223.13**	0.52	**223.13**	5.00	**223.13**	0.74
a4-24	318.21	30.58	**316.65**	0.90	319.37	5.00	**316.65**	1.67
a4-32	430.07	120.00	**397.87**	1.19	401.97	5.00	**397.87**	1.91
a4-40	NA	120.00	479.02	1.91	471.72	5.00	**467.72**	2.31
a4-48	NA	120.00	582.22	2.74	579.71	5.00	**575.62**	3.25
a5-40	447.63	120.00	424.26	1.63	420.20	5.00	**418.75**	3.48
a5-50	NA	120.00	603.24	2.64	593.71	5.00	**589.61**	4.56

Table 6.4 Results on Limmer large instances with $\gamma = 0.7$

Ins.	Limmer (2023)		Proposed VNS (Comp)			Proposed VNS (15 min t.l)	
	Obj	Time (min)	Obj	Time (min)	Gap (%)	Obj	Gap (%)
a180-3600	29,177.96	15.00	27,530.74	16.25	-5.65	28,072.12	-3.79
a200-4000	32,013.56	15.00	30,118.41	20.14	-5.92	31,135.36	-2.74
a220-4400	35,259.06	15.00	32,920.57	23.53	-6.63	34,742.76	-1.46
a240-4800	38,270.98	15.00	35,682.44	27.04	-6.76	38,118.58	-0.40
a260-5200	41,472.11	15.00	38,969.93	29.99	-6.03	41,454.19	-0.04
Average		**15.00**		**23.39**	**-6.20**		**-1.69**

As shown in the results, the proposed algorithm either matches or improves upon the best-known solutions from the literature in most instances. This performance is particularly notable in large-scale instances (Table 6.4), where the algorithm achieved a gap of 6.20% under the complete time limit and 1.69% within a 15-minute time limit when compared to the results reported by Limmer [97].

6.5.3 Case 2 - concave piecewise linear charging function

In the second case, we evaluate the performance of the proposed algorithm when electric vehicles are charged using a concave piecewise linear charging function. Unlike the first case, where a constant charging rate is assumed, this approach reflects a more realistic charging behavior in which the rate of energy transfer decreases as the battery becomes fuller. Notably, this is the first time in the literature that such a charging function is considered within this context. It is important to highlight that the objective function remains unchanged across both cases, focusing on the minimization of total travel time and ride time. This comparison allows us to assess the algorithm's effectiveness under different charging dynamics.

In the concave piecewise linear charging function, 80% of the battery capacity is charged at a constant rate, similar to the linear function mode (0.055 kWh of energy per minute). However, the remaining 20% is charged in two successive linear stages, each with a reduced charging speed, resulting in a concave overall profile [118]. The total time required to achieve a full charge under the concave piecewise linear function is longer than that of the fully linear model. This adjustment in the charging function provides valuable insight into the algorithm's robustness when exposed to more realistic and complex energy replenishment constraints. The results of the proposed algorithm on the large-scale instances from Limmer [97] and Dong et al. [1], under Case 2, are presented in Tables 6.5 and 6.6, respectively.

Table 6.5 Results on Limmer large instances with concave piecewise linear charging function ($\gamma = 0.7$)

Ins.	Linear charging		Concave piecewise linear charging	
	Obj	Time (min)	Obj	Time (min)
a180-3600	27,530.74	16.25	28,168.05	17.11
a200-4000	30,118.41	20.14	30,791.76	22.25
a220-4400	32,920.57	23.53	33,815.90	26.39
a240-4800	35,682.44	27.04	36,896.59	29.84
a260-5200	38,969.93	29.99	40,183.71	34.21

An analysis of the results reveals that the use of a concave piecewise linear charging function, as introduced in Case 2, leads to an average increase of 2.8% and 3.3% in the objective function value compared to the base case with a constant charging rate for Limmer and

Table 6.6 Results on Dong et al. large and very large instances with concave piecewise linear charging function ($\gamma = 0.4$)

Ins.	Linear charging		Concave piecewise linear charging	
	Obj	Time (min)	Obj	Time (min)
s100-1000-1	5,529.01	11.91	5,690.81	12.52
s100-1000-2	5,408.37	10.72	5,570.30	10.87
s100-1000-3	5,722.13	11.55	5,927.99	12.07
s150-1500-1	7,788.22	16.07	8,004.30	17.04
s150-1500-2	7,853.97	14.58	8,139.36	14.98
s150-1500-3	7,762.76	15.89	8,034.68	16.45
s200-2000-1	10,354.22	18.76	10,710.39	18.94
s200-2000-2	10,217.30	17.49	10,531.01	17.64
s200-2000-3	10,039.47	19.22	10,358.29	19.68
s900-10000-1	82,681.15	94.84	85,708.43	98.34
s900-10000-2	82,042.72	97.22	85,081.85	99.13
s900-10000-3	82,798.34	88.54	85,255.31	94.04

Dong et al. instances, respectively. This increase is primarily attributed to the longer charging times required under the concave profile, particularly during the final stages of battery replenishment. Despite this added complexity, the proposed algorithm maintained a comparable computational time across both cases, demonstrating its robustness and efficiency in handling more realistic charging dynamics without compromising performance. These findings underscore the importance of incorporating practical energy consumption behaviors into model formulations while preserving computational tractability.

6.5.4 Case 3 - multiple charging infrastructure types

In the third case, the charging process remains a concave piecewise linear function, similar to Case 2. However, a key extension is introduced: the presence of two types of charging infrastructure—fast and slow chargers. This setting reflects a more realistic urban mobility scenario, where electric vehicles may have access to heterogeneous charging options. The slow charger follows the same concave piecewise linear charging profile used in Case 2, whereas the fast charger is designed to significantly reduce the total charging time, even compared to the constant-rate linear charging model, as illustrated in Figure 6.4. This distinction enables the evaluation of the algorithm's performance under diverse charging conditions and its ability to leverage infrastructure heterogeneity to enhance routing and charging decisions.

To accurately account for the presence of multiple charger types, the insertion model in this case must be updated to reflect the specific characteristics of each charging infrastructure. In particular, the model must identify the type of charger—fast or slow—used at each station, as this directly influences the associated charging time and energy replenishment profile. Each

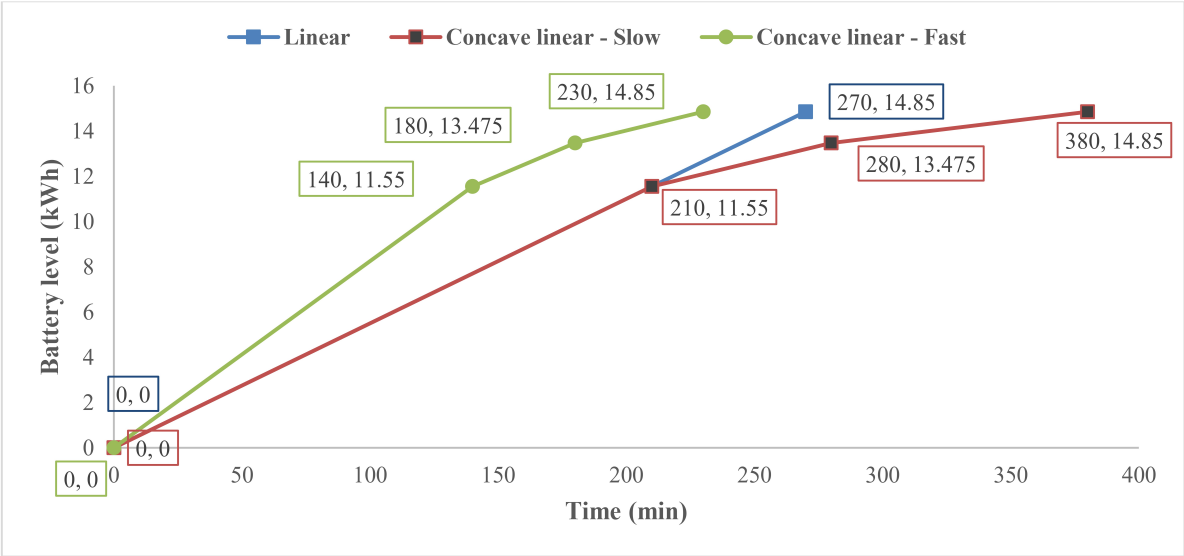


Figure 6.4 Charging process for linear and two types of concave piecewise linear charging functions

charging station is equipped with only one type of charger, and the station index is required to determine the applicable charging function.

In the Limmer large instances, three charging stations are considered, among which two are equipped with slow chargers and one with a fast charger. In Dong et al. instances, the problem setup includes four charging stations, among which two are assumed to be equipped with fast chargers. The integration of a fast-charging station introduces the potential to reduce total charging time for vehicles routed through this facility. Compared to Case 2, this added flexibility enables more efficient energy replenishment, allowing certain vehicles to return to service more quickly and potentially increasing overall request coverage. By strategically routing vehicles to the fast-charging station when beneficial, the algorithm can exploit infrastructure heterogeneity to improve operational efficiency. The results of the proposed algorithm for Case 3 are reported in Tables 6.7 and 6.8.

Table 6.7 Results on Limmer large instances for Case 3 ($\gamma = 0.7$)

Ins.	Case 2		Case 3	
	Obj	Time (min)	Obj	Time (min)
a180-3600	28,168.05	17.11	27,362.44	17.93
a200-4000	30,791.76	22.25	29,827.98	23.38
a220-4400	33,815.90	26.39	32,703.36	27.75
a240-4800	36,896.59	29.84	36,007.38	30.91
a260-5200	40,183.71	34.21	38,893.81	35.51

Table 6.8 Results on Dong et al. large and very large instances for Case 3 ($\gamma = 0.4$)

Ins.	Case 2		Case 3	
	Obj	Time (min)	Obj	Time (min)
s100-1000-1	5,690.81	12.52	5,501.36	12.74
s100-1000-2	5,570.30	10.87	5,373.39	11.29
s100-1000-3	5,927.99	12.07	5,711.77	12.67
s150-1500-1	8,004.30	17.04	7,754.06	17.91
s150-1500-2	8,139.36	14.98	7,891.02	15.29
s150-1500-3	8,034.68	16.45	7,740.15	17.16
s200-2000-1	10,710.39	18.94	10,358.84	19.17
s200-2000-2	10,531.01	17.64	10,164.22	17.92
s200-2000-3	10,358.29	19.68	10,021.86	20.04
s900-10000-1	85,708.43	98.34	82,643.18	99.42
s900-10000-2	85,081.85	99.13	82,059.47	99.81
s900-10000-3	85,255.31	94.04	82,485.51	95.62

As expected, the introduction of fast chargers had a notable impact on solution quality. The number of visits to the charging station equipped with the fast charger increased significantly, by an average of 18% across all large instances, indicating that the algorithm effectively prioritized the use of more efficient charging infrastructure. This strategic adjustment contributed to a meaningful reduction in the overall objective function value, with improvements exceeding 2.8% for the Limmer instances and 3.4% for the Dong et al. instances when compared to Case 2. These results highlight the operational benefits of incorporating heterogeneous charging infrastructure into the problem setting and underscore the responsiveness of the proposed algorithm to such enhancements.

6.5.5 Case 4 - time-of-use electricity pricing

In the fourth case, the objective function is expanded to include charging-related costs, providing a more realistic representation of electric vehicle operations. Two cost structures are considered. In the first scenario, a constant cost rate is applied throughout the planning horizon, with 1$ per minute for travel and waiting time (as in the base case) and an additional 0.9$ per minute as a fixed charging cost. In the second scenario, a time-of-use (TOU) electricity pricing scheme is introduced to reflect realistic charging costs based on temporal electricity tariffs in Canada. Under this structure, charging costs vary across three time periods—off-peak, mid-peak, and on-peak—as illustrated in Figure 6.5. In this case, similar to the previous two cases, the charging process is modeled using a concave piecewise linear charging function, and a single type of charger—a slow charger—is assumed throughout the network. This formulation allows for the assessment of the algorithm's responsiveness to temporal cost variations and its ability to exploit lower-cost charging windows while maintaining

service quality.

The proposed algorithm was applied to each of the two cost structures described above for the Limmer large-size instances, and the corresponding results are presented in Table 6.9. As observed, incorporating a fixed charging cost throughout the planning horizon leads to higher total operating costs. In contrast, the TOU electricity pricing scheme enables the algorithm to schedule charging activities more cost-effectively by leveraging off-peak periods. Consequently, this dynamic pricing approach results in a notable reduction in charging costs (10.24% on average) and contributes to a decrease in the total objective value by an average of 2.3% across the tested instances. This highlights the importance of accounting for temporal variations in electricity prices when planning charging strategies in EV routing problems.

Table 6.9 Influence of TOU electricity pricing policy on costs - Limmer Instances ($\gamma = 0.7$)

Ins.	Constant Rate			TOU electricity pricing policy			Gap (%)
	Travel and ride time cost	Charging cost	Total cost	Travel and ride time cost	Charging cost	Total cost	
a180-3600	40,574.28	22,761.97	63,336.25	41,324.49	20,334.68	61,659.17	-2.65
a200-4000	44,357.90	24,613.88	68,971.78	45,237.43	22,044.40	67,281.84	-2.45
a220-4400	48,514.84	26,814.66	75,329.50	49,590.50	24,110.60	73,701.10	-2.16
a240-4800	53,080.28	29,397.21	82,477.49	54,285.05	26,453.39	80,738.44	-2.11
a260-5200	57,911.91	32,228.53	90,140.44	59,229.57	28,998.64	88,228.21	-2.12

To enable a fair comparison with the results reported by Dong et al. [1], the TOU electricity pricing for their benchmark instances was adapted based on Shenzhen's current electricity pricing policy. It is important to highlight that, in Dong et al. [1], the charging process was modeled using a linear function. Accordingly, to ensure consistency in this comparative analysis, the same linear charging function was adopted when implementing the proposed algorithm on these instances (Table 6.10). This alignment allows for isolating the effect of incorporating TOU-based charging costs in the objective function while maintaining compatibility with the assumptions made in the original benchmark study.

As shown in Table 6.10, the proposed algorithm outperforms the approach of Dong et al. [1] in solving large and very large instances. Specifically, it achieves improved objective function values with an average gap of -1.52%, while also significantly reducing the computational effort, with an average solution time reduction of approximately 85%. These results highlight the efficiency and effectiveness of the proposed approach, particularly in handling computationally demanding real-world instances.

6.5.6 Case 5 - charging station capacity

In the literature on the E-DARP, most studies focus on optimizing vehicle routes and scheduling while ensuring sufficient battery levels through strategically placed charging stations.

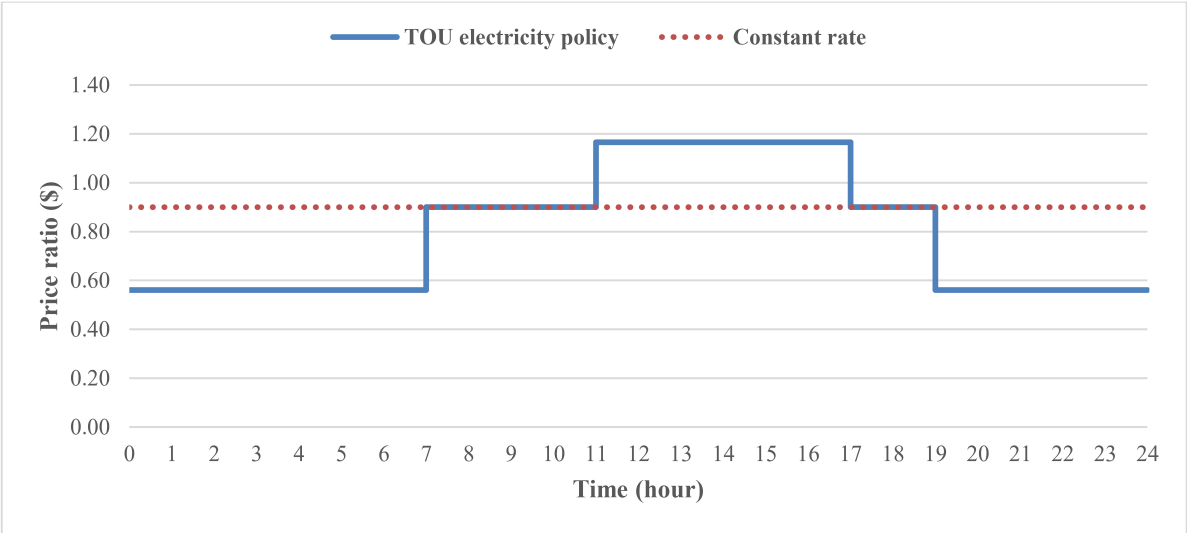


Figure 6.5 TOU electricity pricing policy

However, a critical yet often overlooked factor is the limited capacity of charging stations, which significantly impacts real-world feasibility. Ignoring this constraint can lead to unrealistic solutions where multiple vehicles are scheduled to charge at the same station simultaneously, exceeding its available chargers. This can cause severe delays, forcing vehicles to wait for charging availability or deviate from their planned routes to find alternative stations, ultimately increasing total operational costs and service delays. Incorporating charging station capacity constraints ensures that solutions remain practical and implementable, leading to more efficient fleet operations and reducing congestion at high-demand locations. Addressing this issue is particularly important in large-scale E-DARP instances, where multiple vehicles rely on shared charging infrastructure, making capacity-aware scheduling a crucial element for real-world applicability.

The problem of optimally assigning charging stations to vehicles while considering station capacity constraints can be formulated as a large-scale mixed-integer linear programming (MILP) model. In this formulation, binary decision variables would determine whether a vehicle charges at a given station and at what time, while continuous variables would capture charging durations and battery levels. Constraints would ensure that (1) vehicles charge within their available time windows, (2) charging stations do not exceed their capacity at any given time, and (3) vehicles complete their routes with sufficient battery levels. However, solving this MILP directly for large instances would be computationally intractable due to the exponential number of possible charging assignments and time conflicts.

To overcome the computational complexity of incorporating charging station capacity con-

Table 6.10 Comparison of the proposed algorithm with Dong et al. [1] under TOU-based electricity pricing ($\gamma = 0.4$)

Ins.	Dong et al. (2025)		Proposed VNS		
	Best Obj	Time (min)	Obj	Time (min)	Gap(%)
s100-1000-1	7,655.50	66.08	7,607.65	12.15	-0.63
s100-1000-2	7,480.48	39.00	7,441.66	11.09	-0.52
s100-1000-3	7,932.00	59.80	7,873.38	12.02	-0.74
s150-1500-1	10,845.49	141.22	10,716.21	16.23	-1.19
s150-1500-2	10,952.56	114.33	10,806.68	14.97	-1.33
s150-1500-3	10,849.73	152.78	10,681.18	16.41	-1.55
s200-2000-1	14,513.15	384.55	14,246.91	19.23	-1.83
s200-2000-2	14,334.24	281.67	14,058.51	17.75	-1.92
s200-2000-3	14,064.50	390.27	13,813.82	19.45	-1.78
s900-10000-1	115,996.28	431.97	113,765.27	97.37	-1.92
s900-10000-2	115,401.23	438.67	112,886.82	98.45	-2.18
s900-10000-3	116,995.84	427.29	113,926.51	89.84	-2.62
Average		**243.97**		**35.41**	**-1.52**

straints into large-scale E-DARP instances, we propose a sequential assignment heuristic that ensures capacity-aware charging while maintaining computational efficiency. The heuristic follows a structured process:

1. **Sort** vehicles based on the number of potential charging stations along their planned routes. Vehicles with fewer available charging options are prioritized to minimize the risk of infeasibility.

2. **For** each vehicle in the sorted list:

   - **Solve** the insertion model to determine the optimal charging station and time slot, considering the current charging station occupation schedule.
   - **Assign** the selected charging time to the vehicle and **update** the station's occupation schedule accordingly.
   - **If** the insertion model is infeasible due to lack of available capacity, attempt to assign the vehicle to the next closest feasible station and its available time window.

This heuristic allows vehicles to be assigned sequentially in a conflict-aware manner. By continuously updating station occupation schedules, it ensures that no more vehicles are scheduled to charge than the number of chargers available at any given time. If a conflict arises, the algorithm dynamically explores alternative feasible assignments. The procedure is applied during the initial solution construction phase, ensuring that generated routes already respect charging station capacity constraints. Additionally, during the improvement phase of

the algorithm, whenever routes are modified, Step 2 is re-executed for the affected vehicles to maintain the feasibility of the solution concerning charging infrastructure limitations.

The results for this case, applied to the Limmer and Dong et al. instances, are presented in Tables 6.11 and 6.12, respectively. The minimum required capacity (i.e., number of chargers) for each charging station is estimated by rounding up the ratio of the total required charging time in Case 2 (i.e., when there is no limitation on the number of chargers) to the total available operational time of stations per day. As expected, incorporating capacity limitations at charging stations resulted in the rerouting of certain vehicle paths, leading to an increase in the total operational cost. Specifically, the cost increased by 3.75% for the Limmer instances and 4.41% for the Dong et al. instances compared to the previous case without capacity constraints. Additionally, the algorithm's computational time increased by 18% and 11%, respectively, reflecting the added complexity introduced by considering charging station availability.

Table 6.11 Results on Limmer large instances for Case 5 ($\gamma = 0.7$)

Ins.	Case 2 (without station capacity)		Case 5 (with station capacity)			Case 5 (with increasing capacity)			
	Obj	Time (min)	Minimum capacity	Obj	Time (min)	0	+1	+2	+3
a180-3600	28,168.05	17.11	6	29,451.38	22.38	4.56%	2.99%	1.41%	0.46%
a200-4000	30,791.76	22.25	7	31,719.63	27.29	3.01%	1.95%	0.96%	0.41%
a220-4400	33,815.90	26.39	7	35,307.64	29.41	4.41%	2.73%	1.32%	0.51%
a240-4800	36,896.59	29.84	8	38,167.92	36.12	3.45%	2.04%	1.00%	0.43%
a260-5200	40,183.71	34.21	9	41,525.37	43.76	3.34%	2.19%	1.13%	0.52%

Table 6.12 Results on Dong et al. large and very large instances for Case 5 ($\gamma = 0.4$)

Ins.	Case 2 (without station capacity)		Case 5 (with station capacity)			Case 5 (with increasing capacity)			
	Obj	Time (min)	Minimum capacity	Obj	Time (min)	0	+1	+2	+3
s100-1000-1	5,690.81	12.52	1	5,963.64	13.40	4.79%	3.01%	1.14%	0.29%
s100-1000-2	5,570.30	10.87	1	5,758.46	12.81	3.38%	1.95%	0.80%	0.22%
s100-1000-3	5,927.99	12.07	1	6,126.02	12.92	3.34%	1.91%	0.74%	0.24%
s150-1500-1	8,004.30	17.04	1	8,367.61	18.79	4.54%	3.02%	1.24%	0.32%
s150-1500-2	8,139.36	14.98	1	8,428.04	18.26	3.55%	2.03%	0.76%	0.27%
s150-1500-3	8,034.68	16.45	1	8,404.79	17.87	4.61%	3.01%	1.15%	0.30%
s200-2000-1	10,710.39	18.94	2	11,133.10	20.77	3.95%	2.36%	0.92%	0.32%
s200-2000-2	10,531.01	17.64	2	11,005.26	18.99	4.50%	2.87%	1.12%	0.34%
s200-2000-3	10,358.29	19.68	2	10,778.05	22.76	4.05%	2.36%	0.90%	0.31%
s900-10000-1	85,708.43	98.34	3	90,211.58	106.57	5.25%	3.48%	1.34%	0.45%
s900-10000-2	85,081.85	99.13	3	89,834.22	109.74	5.59%	3.39%	1.39%	0.51%
s900-10000-3	85,255.31	94.04	3	89,883.71	104.52	5.43%	3.47%	1.34%	0.48%

To evaluate the sensitivity of the objective value to charging station capacity, the minimum estimated number of chargers was incrementally increased by 1, 2, and 3 units per station. The resulting percentage gaps in objective value between Case 5 (with capacity constraints) and Case 2 (without capacity constraints) are presented in Tables 6.11 and 6.12 for the Limmer and Dong et al. instances. As observed, increasing the number of chargers leads to a

consistent reduction in the gap, indicating improved flexibility in utilizing charging stations and reduced operational costs. These results clearly indicate that enhancing the capacity of charging stations can effectively reduce operational costs by minimizing waiting times and rerouting due to congestion. However, it is also important to balance the infrastructure investment cost with the operational savings.

6.6 Conclusion

This paper presents a novel and comprehensive approach to the E-DARP, addressing several limitations in the existing literature by incorporating key electric vehicle features often overlooked in prior studies. In contrast to traditional models that assume homogeneous batteries, fully charged vehicles, and linear charging functions, our problem integrates heterogeneous charging infrastructure, partial charging, and time-dependent charging pricing policies, as well as a concave piecewise linear charging function to more accurately reflect real-world charging dynamics. Importantly, we are the first to explicitly incorporate charging station capacity constraints into the E-DARP, which significantly enhances the practical relevance of the problem formulation. To effectively solve this complex and large-scale problem, we proposed a tailored VNS algorithm enriched with adaptive neighborhood structures, charging station insertion strategies, and a mixed-integer programming component for optimizing charging decisions. Computational experiments on extensive benchmark and real-world-based datasets, including instances with up to 10,000 requests, demonstrate the algorithm's scalability and ability to produce high-quality solutions. The results show that our method consistently outperforms baseline approaches and successfully handles each of the new challenges introduced. Overall, this study contributes both theoretically and practically to the field of electric mobility and on-demand transport systems by proposing a realistic, flexible, and scalable solution methodology. Future research may extend this work by considering stochastic travel times, dynamic demand environments, or integration with renewable energy resources in the charging process.

Acknowledgements

We gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada under the Discovery grants 2023-03791 and 2020-06903.

CHAPTER 7 GENERAL DISCUSSION

This dissertation has explored a sequence of increasingly complex and operationally motivated variants of the DARP, with a particular emphasis on scalability, realism, and energy-aware planning. Conducted in close collaboration with GIRO Inc., a leading provider of software for transit and paratransit systems, the research has been grounded in real-world operational contexts. The company's data and planning requirements provided both the foundation and the validation environment for the models and algorithms proposed throughout this work. Collectively, this thesis aims to bridge the persistent gap between the theoretical development of routing algorithms and the practical needs of large-scale, demand-responsive transportation systems.

In Chapter 4, a practical version of the DARP that incorporates user and vehicle heterogeneity, contract-based cost and regulatory constraints, route duration limits, break rules, and capacity constraints was addressed. This variant, reflective of GIRO's planning environment, was formulated as a set-partitioning problem and solved using a tailored BPC algorithm. The solution approach leveraged a column generation framework supported by a sophisticated labeling algorithm capable of handling the full set of operational constraints, including contract-specific rules and resource types. Computational results on real-world data demonstrated the algorithm's ability to optimally solve instances with up to 849 requests and over 70 heterogeneous vehicles—representing the largest known instances of this kind solved to proven optimality. This work served a dual purpose: it benchmarked the performance of GIRO's industrial heuristics and validated the practical relevance of exact methods for moderate-scale applications.

Chapter 5 shifted the focus toward computational scalability, addressing the challenges posed by very large-scale DARP instances often found in real-world systems. A VNS algorithm was developed to manage problems involving thousands of transportation requests and hundreds of vehicles. The method integrated a linear relaxation-based constructive phase followed by iterative refinement through a diverse set of neighborhoods. To enhance solution quality, the heuristic was further reinforced with embedded MIP components for local optimization. Tested on very large-scale datasets ranging from 2,932 to over 10,500 requests, the algorithm consistently produced high-quality solutions within competitive computational times—often under one hour. For medium-sized cases tested in Chapter 4, the VNS achieved near-optimal solutions with small optimality gaps (below 0.5%) while significantly reducing computation time compared to the exact method. This contribution fully addressed the second research

objective, demonstrating how heuristic methods can scale to meet real-world operational demands while preserving the modeling complexity introduced in earlier work.

The final contribution, presented in Chapter 6, extended the DARP framework to accommodate EV fleets, leading to the formulation of the E-DARP. This problem introduced new layers of complexity, including battery constraints, charging operations, time-of-use electricity pricing, and heterogeneous charging station capacities. A key innovation was the integration of concave piecewise linear charging functions and the explicit modeling of charging station capacity—elements rarely addressed in existing literature. The VNS algorithm developed in Chapter 5 was adapted to include energy-aware routing logic, charging station insertion strategies, and MIP-based modules for managing battery dynamics and recharging decisions. Computational experiments on industrial-scale instances (up to 10,000 requests) confirmed the robustness and scalability of the method, demonstrating its capacity to produce feasible and energy-efficient solutions for large electric vehicle fleets. Chapter 6 fulfilled the third research objective by offering a novel and practical approach to energy-aware DARP optimization.

In summary, the dissertation provides a unified and scalable methodological framework that advances both the modeling and solution capabilities for conventional and electric DARP variants. It offers exact and heuristic tools that are not only theoretically sound but also validated on real-world datasets of unprecedented size and complexity. Through its three contributions, the thesis has systematically addressed the research objectives: delivering an exact method for realistic small- to medium-sized problems, proposing a heuristic for very large-scale systems, and extending the framework to integrate electric mobility constraints. These outcomes contribute directly to the advancement of decision-support systems in public and on-demand transportation planning, providing tools that are both operationally viable and adaptable to future sustainability challenges.

CHAPTER 8 CONCLUSION

This dissertation has addressed the DARP and its EV variant (E-DARP) by developing a set of advanced optimization algorithms capable of tackling realistic and large-scale transportation planning scenarios. Through a combination of exact and heuristic methods, the research has responded to the increasing complexity of modern demand-responsive mobility systems. A BPC algorithm was proposed to solve practical DARP instances to optimality, while a VNS framework was designed and extended to efficiently handle very large-scale DARP and E-DARP instances. These contributions were validated on industrial datasets provided by GIRO Inc., demonstrating both theoretical rigor and practical relevance.

Despite these advances, the research is subject to several limitations that open the door to future investigation. First, the exact algorithm presented in Chapter 4, while capable of solving realistic instances, remains computationally intractable for larger datasets. Its scalability is constrained by the exponential growth of the solution space and the complexity of the pricing subproblem. While heuristics mitigate this issue, future work may focus on developing matheuristics that combine the strengths of heuristic exploration with the theoretical guarantees of exact algorithms. Embedding heuristic BPC components within a large-scale framework could enable the solution of large problems for better trade-offs between solution quality and runtime.

Second, although the proposed VNS algorithm demonstrated strong performance across large-scale instances, its success relies in part on the quality of the initial solution and the design of the neighborhood structures. Further research could investigate adaptive or learning-based neighborhood selection mechanisms to improve convergence behavior. Additionally, exploring parallel and distributed versions of the algorithm may enhance its scalability and suitability for real-time or large-fleet deployment environments.

Third, the E-DARP formulation introduced in Chapter 6 integrates a range of realistic features, such as nonlinear charging behavior, station capacity, and time-of-use pricing. However, some simplifying assumptions were made—for example, deterministic travel times, fixed energy consumption rates, and static request sets. Future extensions could consider stochastic elements, such as uncertain demand or variable energy consumption based on traffic conditions, as well as dynamic and online variants of the E-DARP, where requests arrive in real time and charging station availability may fluctuate.

Lastly, while charging station insertion and scheduling are explicitly modeled, vehicle routing decisions are not yet co-optimized with infrastructure planning decisions, such as the

deployment or expansion of charging facilities. This restricts the long-term applicability of the model for strategic planning. Combining routing decisions with infrastructure planning represents a valuable long-term direction. Joint optimization of electric fleet routing and charging station placement could lead to more effective policies for the deployment of smart, resilient, and sustainable urban transport networks.

REFERENCES

[1] H. Dong, Z. Luo, N. Huang, H. Hu, and H. Qin, "The electric vehicle dial-a-ride problem: Integrating ride-sharing and time-of-use electricity pricing," *Transportation Research Part E: Logistics and Transportation Review*, vol. 194, p. 103946, 2025.

[2] N. Wilson, J. Sussman, H. Wong, and B. Higonnet, "Scheduling algorithms for dial-a-ride systems," Urban Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA, Technical Report, 1971.

[3] J. F. Cordeau and G. Laporte, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research*, vol. 153, pp. 29–46, 2007.

[4] S. Ropke and J. F. Cordeau, "Branch and cut and price for the pickup and delivery problem with time windows," *Transportation Science*, vol. 43, no. 3, pp. 267–286, 2009.

[5] O. B. G. Madsen, H. F. Ravn, and J. M. Rygaard, "A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives," *Annals of Operations Research*, vol. 60, no. 1, pp. 193–208, 1995.

[6] E. Melachrinoudis, A. B. Ilhan, and H. Min, "A dial-a-ride problem for client transportation in a health-care organization," *Computers & Operations Research*, vol. 34, no. 3, pp. 742–759, 2007.

[7] P. Toth and D. Vigo, "Fast local search algorithms for the handicapped persons transportation problem," in *Meta-Heuristics*, I. H. Osman and J. P. Kelly, Eds. Boston, MA: Springer, 1996, pp. 677–690.

[8] J. Paquette, J.-F. Cordeau, G. Laporte, and M. M. B. Pascoal, "Combining multicriteria analysis and tabu search for dial-a-ride problems," *Transportation Research Part B: Methodological*, vol. 52, pp. 1–16, 2013.

[9] M. W. P. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995.

[10] P. Detti, F. Papalini, and G. Z. M. de Lara, "A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare," *Omega*, vol. 70, pp. 1–14, 2017.

[11] U. Ritzinger, J. Puchinger, and R. F. Hartl, "Dynamic programming based metaheuristics for the dial-a-ride problem," *Annals of Operations Research*, vol. 236, no. 2, pp. 341–358, 2016.

[12] S. N. Parragh, "Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 5, pp. 912–930, 2011.

[13] S. N. Parragh, J.-F. Cordeau, K. F. Doerner, and R. F. Hartl, "Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints," *OR Spectrum*, vol. 34, no. 3, pp. 593–633, 2012.

[14] J. Zhao, M. Poon, Z. Zhang, and R. Gu, "Adaptive large neighborhood search for the time-dependent profitable dial-a-ride problem," *Computers & Operations Research*, vol. 147, p. 105938, 2022.

[15] C. Bongiovanni, M. Kaspi, and N. Geroliminis, "The electric autonomous dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 122, pp. 436–456, 2019.

[16] S. N. Parragh, K. F. Doerner, R. F. Hartl, and X. Gandibleux, "A heuristic two-phase solution approach for the multi-objective dial-a-ride problem," *Networks: An International Journal*, vol. 54, no. 4, pp. 227–242, 2009.

[17] J.-F. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 37, no. 6, pp. 579–594, 2003.

[18] T. Gschwind and S. Irnich, "Effective handling of dynamic time windows and its application to solving the dial-a-ride problem," *Transportation Science*, vol. 49, no. 2, pp. 335–354, 2015.

[19] Z. Luo, M. Liu, and A. Lim, "A two-phase branch-and-price-and-cut for a dial-a-ride problem in patient transportation," *Transportation Science*, vol. 53, no. 1, pp. 113–130, 2019.

[20] R. Chevrier, A. Liefooghe, L. Jourdan, and C. Dhaenens, "Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport," *Applied Soft Computing*, vol. 12, no. 4, pp. 1247–1258, 2012.

[21] T. Garaix, C. Artigues, D. Feillet, and D. Josselin, "Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation," *Computers & Operations Research*, vol. 38, no. 10, pp. 1435–1442, 2011.

[22] S. N. Parragh, J. P. de Sousa, and B. Almada-Lobo, "The dial-a-ride problem with split requests and profits," *Transportation Science*, vol. 49, no. 2, pp. 311–334, 2015.

[23] A. Lim, Z. Zhang, and H. Qin, "Pickup and delivery service with manpower planning in hong kong public hospitals," *Transportation Science*, vol. 51, no. 2, pp. 688–705, 2017.

[24] V. Pimenta, A. Quilliot, H. Toussaint, and D. Vigo, "Models and algorithms for reliability-oriented dial-a-ride with autonomous electric vehicles," *European Journal of Operational Research*, vol. 257, no. 2, pp. 601–613, 2017.

[25] T. Y. Hu, G. C. Zheng, and T. Y. Liao, "Multi-objective model for dial-a-ride problems with vehicle speed considerations," *Transportation Research Record*, vol. 2673, no. 11, pp. 161–171, 2019.

[26] R. M. Jorgensen, J. Larsen, and K. B. Bergvinsdottir, "Solving the dial-a-ride problem using genetic algorithms," *Journal of the Operational Research Society*, vol. 58, no. 10, pp. 1321–1331, 2007.

[27] D. Kirchler and R. W. Calvo, "A granular tabu search algorithm for the dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 56, pp. 120–135, 2013.

[28] M. Schilde, K. F. Doerner, and R. F. Hartl, "Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports," *Computers & Operations Research*, vol. 38, no. 12, pp. 1719–1730, 2011.

[29] ——, "Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem," *European Journal of Operational Research*, vol. 238, no. 1, pp. 18–28, 2014.

[30] O. Tellez, S. Vercraene, F. Lehuédé, O. Péton, and T. Monteiro, "The time-consistent dial-a-ride problem," *Networks*, vol. 79, no. 4, pp. 452–478, 2022.

[31] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "Variable neighborhood search for the dial-a-ride problem," *Computers & Operations Research*, vol. 37, no. 6, pp. 1129–1138, 2010.

[32] S. N. Parragh and V. Schmid, "Hybrid column generation and large neighborhood search for the dial-a-ride problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 490–497, 2013.

[33] M. Chassaing, C. Duhamel, and P. Lacomme, "An els-based approach with dynamic probabilities management in local search for the dial-a-ride problem," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 119–133, 2016.

[34] C. E. Cortés, M. Matamala, and C. Contardo, "The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method," *European Journal of Operational Research*, vol. 200, no. 3, pp. 711–724, 2010.

[35] R. Masson, F. Lehuédé, and O. Péton, "The dial-a-ride problem with transfers," *Computers & Operations Research*, vol. 41, pp. 12–23, 2014.

[36] J. Schönberger, "Scheduling constraints in dial-a-ride problems with transfers: a metaheuristic approach incorporating a cross-route scheduling procedure with postponement opportunities," *Public Transport*, vol. 9, no. 1, pp. 243–272, 2017.

[37] C. H. Häll, H. Andersson, J. T. Lundgren, and P. Värbrand, "The integrated dial-a-ride problem," *Public Transport*, vol. 1, no. 1, pp. 39–54, 2009.

[38] M. Posada, H. Andersson, and C. H. Häll, "The integrated dial-a-ride problem with timetabled fixed route service," *Public Transport*, vol. 9, no. 1, pp. 217–241, 2017.

[39] T. Grinshpoun, E. Shufan, H. Ilani, V. Levit, and H. Brama, "Effective pruning heuristics for the fixed route dial-a-ride problem," in *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT*, vol. 1, 2020.

[40] Z. Zhang, M. Liu, and A. Lim, "A memetic algorithm for the patient transportation problem," *Omega*, vol. 54, pp. 60–71, 2015.

[41] M. Liu, Z. Luo, and A. Lim, "A branch-and-cut algorithm for a realistic dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 81, pp. 267–288, 2015.

[42] M. A. Masmoudi, M. Hosny, E. Demir, K. N. Genikomsakis, and N. Cheikhrouhou, "The dial-a-ride problem with electric vehicles and battery swapping stations," *Transportation Research Part E: Logistics and Transportation Review*, vol. 118, pp. 392–420, 2018.

[43] Y. Molenbruch, K. Braekers, O. Eisenhandler, and M. Kaspi, "The electric dial-a-ride problem on a fixed circuit," *Transportation Science*, vol. 57, no. 3, pp. 594–612, 2023.

[44] E. Hyytiä, S. Aalto, A. Penttinen, and R. Sulonen, "A stochastic model for a vehicle in a dial-a-ride system," *Operations Research Letters*, vol. 38, no. 5, pp. 432–435, 2010.

[45] S. C. Ho and D. Haugland, "Local search heuristics for the probabilistic dial-a-ride problem," *OR Spectrum*, vol. 33, no. 4, pp. 961–988, 2011.

[46] G. Heilporn, J. F. Cordeau, and G. Laporte, "An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays," *Discrete Applied Mathematics*, vol. 159, no. 9, pp. 883–895, 2011.

[47] C. H. Häll, J. T. Lundgren, and S. Voss, "Evaluating the performance of a dial-a-ride service using simulation," *Public Transport*, vol. 7, pp. 139–157, 2015.

[48] G. Berbeglia, J. F. Cordeau, and G. Laporte, "A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem," *INFORMS Journal on Computing*, vol. 24, no. 3, pp. 343–355, 2012.

[49] N. Marković, R. Nair, P. Schonfeld, E. Miller-Hooks, and M. Mohebbi, "Optimizing dial-a-ride services in Maryland: benefits of computerized routing and scheduling," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 156–165, 2015.

[50] D. O. Santos and E. C. Xavier, "Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6728–6737, 2015.

[51] H. R. Sayarshad and J. Y. J. Chow, "A scalable non-myopic dynamic dial-a-ride and pricing problem," *Transportation Research Part B: Methodological*, vol. 81, pp. 539–554, 2015.

[52] H. R. Sayarshad and H. O. Gao, "A scalable non-myopic dynamic dial-a-ride and pricing problem for competitive on-demand mobility systems," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 192–208, 2018.

[53] X. Liang, G. H. de Almeida Correia, K. An, and B. van Arem, "Automated taxis' dial-a-ride problem with ride-sharing considering congestion-based dynamic travel times," *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 260–281, 2020.

[54] C. Paquay, Y. Crama, and T. Pironet, "Recovery management for a dial-a-ride system with real-time disruptions," *European Journal of Operational Research*, vol. 280, no. 3, pp. 953–969, 2020.

[55] A. L. Souza, M. Bernardo, P. H. Penna, J. Pannek, and M. J. Souza, "Bi-objective optimization model for the heterogeneous dynamic dial-a-ride problem with no rejects," *Optimization Letters*, vol. 16, no. 1, pp. 355–374, 2022.

[56] Z. Xiang, C. Chu, and H. Chen, "The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments," *European Journal of Operational Research*, vol. 185, no. 2, pp. 534–551, 2008.

[57] M. Maalouf, C. A. MacKenzie, S. Radakrishnan, and M. Court, "A new fuzzy logic approach to capacitated dynamic dial-a-ride problem," *Fuzzy Sets and Systems*, vol. 255, pp. 30–40, 2014.

[58] D. Muñoz-Carpintero, D. Sáez, C. E. Cortés, and A. Núñez, "A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach," *Transportation Science*, vol. 49, no. 2, pp. 239–253, 2015.

[59] A. Núñez, C. E. Cortés, D. Sáez, B. D. Schutter, and M. Gendreau, "Multiobjective model predictive control for dynamic pickup and delivery problems," *Control Engineering Practice*, vol. 32, pp. 73–86, 2014.

[60] E. Hyytiä, A. Penttinen, and R. Sulonen, "Non-myopic vehicle and route selection in dynamic darp with travel time and workload objectives," *Computers & Operations Research*, vol. 39, no. 12, pp. 3021–3030, 2012.

[61] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[62] Y. Rist and M. A. Forbes, "A new formulation for the dial-a-ride problem," *Transportation Science*, vol. 55, no. 5, pp. 1113–1135, 2021.

[63] D. Gaul, K. Klamroth, and M. Stiglmayr, "Event-based milp models for ridepooling applications," *European Journal of Operational Research*, vol. 301, no. 3, pp. 1048–1063, 2022.

[64] A. Schulz and C. Pfeiffer, "A branch-and-cut algorithm for the dial-a-ride problem with incompatible customer types," *Transportation Research Part E: Logistics and Transportation Review*, vol. 181, p. 103394, 2024.

[65] Y. Qu and J. F. Bard, "A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity," *Transportation Science*, vol. 49, no. 2, pp. 254–270, 2015.

[66] S. Guo, I. Dayarian, J. Li, and X. Qian, "Solving the equity-aware dial-a-ride problem using an exact branch-cut-and-price algorithm," *Transportation Research Part B: Methodological*, vol. 192, p. 103149, 2025.

[67] E. Melachrinoudis and H. Min, "A tabu search heuristic for solving the multi-depot, multi-vehicle, double request dial-a-ride problem faced by a healthcare organisation," *International Journal of Operational Research*, vol. 10, no. 2, pp. 214–239, 2011.

[68] Z. Zang, Q. Tian, and D. Z. Wang, "On-demand transportation system for cross-state abortion travel: A dual dial-a-ride problem," *Transportation Research Part A: Policy and Practice*, vol. 196, p. 104473, 2025.

[69] T. Gschwind and M. Drexl, "Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem," *Transportation Science*, vol. 53, no. 2, pp. 480–491, 2019.

[70] F. Lehuédé, R. Masson, S. N. Parragh, O. Péton, and F. Tricoire, "A multi-criteria large neighbourhood search for the transportation of disabled people," *Journal of the Operational Research Society*, vol. 65, no. 7, pp. 983–1000, 2014.

[71] Y. Molenbruch, K. Braekers, A. Caris, and G. V. Berghe, "Multi-directional local search for a bi-objective dial-a-ride problem in patient transportation," *Computers & Operations Research*, vol. 77, pp. 58–71, 2017.

[72] C. Cubillos, E. Urra, and N. Rodríguez, "Application of genetic algorithms for the darptw problem," *International Journal of Computers Communications & Control*, vol. 4, no. 2, pp. 127–136, 2009.

[73] M. A. Masmoudi, K. Braekers, M. Masmoudi, and A. Dammak, "A hybrid genetic algorithm for the heterogeneous dial-a-ride problem," *Computers & Operations Research*, vol. 81, pp. 1–13, 2017.

[74] K. Braekers, A. Caris, and G. K. Janssens, "Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots," *Transportation Research Part B: Methodological*, vol. 67, pp. 166–186, 2014.

[75] S. Belhaiza, "A hybrid adaptive large neighborhood heuristic for a real-life dial-a-ride problem," *Algorithms*, vol. 12, no. 2, p. 39, 2019.

[76] S. Jain and P. V. Hentenryck, "Large neighborhood search for dial-a-ride problems," in *International Conference on Principles and Practice of Constraint Programming*. Springer, Berlin Heidelberg, 2011, pp. 400–413.

[77] Z. Xiang, C. Chu, and H. Chen, "A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints," *European Journal of Operational Research*, vol. 174, no. 2, pp. 1117–1139, 2006.

[78] S. Muelas, A. LaTorre, and J. M. Pena, "A distributed vns algorithm for optimizing dial-a-ride problems in large-scale scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 110–130, 2015.

[79] ——, "A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5516–5531, 2013.

[80] C. Liu, A. Quilliot, H. Toussaint, and D. Feillet, "A filtering system to solve the large-scale shared autonomous vehicles dial-a-ride problem," *Transportation Research Part C: Emerging Technologies*, vol. 161, p. 104551, 2024.

[81] J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau, "An efficient variable neighborhood search heuristic for very large scale vehicle routing problems," *Computers & Operations Research*, vol. 34, no. 9, pp. 2743–2757, 2007.

[82] M. Qi, W. H. Lin, N. Li, and L. Miao, "A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 248–257, 2012.

[83] F. Arnold, M. Gendreau, and K. Sörensen, "Efficiently solving very large-scale routing problems," *Computers & Operations Research*, vol. 107, pp. 32–42, 2019.

[84] L. Accorsi and D. Vigo, "A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems," *Transportation Science*, vol. 55, no. 4, pp. 832–856, 2021.

[85] K. Zhang, M. Li, J. Wang, Y. Li, and X. Lin, "A two-stage learning-based method for large-scale on-demand pickup and delivery services with soft time windows," *Transportation Research Part C: Emerging Technologies*, vol. 151, p. 104122, 2023.

[86] V. R. Máximo, J. F. Cordeau, and M. C. Nascimento, "Ails-ii: An adaptive iterated local search heuristic for the large-scale capacitated vehicle routing problem," *INFORMS Journal on Computing*, vol. 36, no. 4, pp. 974–986, 2024.

[87] F. Cavaliere, L. Accorsi, D. Laganà, R. Musmanno, and D. Vigo, "An efficient heuristic for very large-scale vehicle routing problems with simultaneous pickup and delivery," *Transportation Research Part E: Logistics and Transportation Review*, vol. 186, p. 103550, 2024.

[88] L. Accorsi and D. Vigo, "Routing one million customers in a handful of minutes," *Computers & Operations Research*, vol. 164, p. 106562, 2024.

[89] L. Grandinetti, F. Guerriero, F. Pezzella, and O. Pisacane, "A pick-up and delivery problem with time windows by electric vehicles," *International Journal of Productivity and Quality Management*, vol. 18, no. 2–3, pp. 403–423, 2016.

[90] D. Goeke, "Granular tabu search for the pickup and delivery problem with time windows and electric vehicles," *European Journal of Operational Research*, vol. 278, no. 3, pp. 821–836, 2019.

[91] M. Soysal, M. Çimen, and S. Belbağ, "Pickup and delivery with electric vehicles under stochastic battery depletion," *Computers & Industrial Engineering*, vol. 146, p. 106512, 2020.

[92] X. Liu, D. Wang, Y. Yin, and T. C. E. Cheng, "Robust optimization for the electric vehicle pickup and delivery problem with time windows and uncertain demands," *Computers & Operations Research*, vol. 151, p. 106119, 2023.

[93] C. Agrali and S. Lee, "The multi-depot pickup and delivery problem with capacitated electric vehicles, transfers, and time windows," *Computers & Industrial Engineering*, vol. 179, p. 109207, 2023.

[94] S. Zhou, D. Zhang, W. Yuan, Z. Wang, L. Zhou, and M. G. Bell, "Pickup and delivery problem with electric vehicles and time windows considering queues," *Transportation Research Part C: Emerging Technologies*, vol. 167, p. 104829, 2024.

[95] C. Bongiovanni, M. Kaspi, J. F. Cordeau, and N. Geroliminis, "A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations," *Transportation Research Part E: Logistics and Transportation Review*, vol. 165, p. 102835, 2022.

[96] Y. Su, N. Dupin, S. N. Parragh, and J. Puchinger, "A column generation approach for the electric autonomous dial-a-ride problem," in *24ème Congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2023)*, Rennes, France, February 2023.

[97] S. Limmer, "Bilevel large neighborhood search for the electric autonomous dial-a-ride problem," *Transportation Research Interdisciplinary Perspectives*, vol. 21, p. 100876, 2023.

[98] Y. Su, N. Dupin, and J. Puchinger, "A deterministic annealing local search for the electric autonomous dial-a-ride problem," *European Journal of Operational Research*, vol. 309, no. 3, pp. 1091–1111, 2023.

[99] M. Bresich, G. R. Raidl, and S. Limmer, "Letting a large neighborhood search for an electric dial-a-ride problem fly: on-the-fly charging station insertion," in *Proceedings of the Genetic and Evolutionary Computation Conference*, Melbourne, Australia, July 2024, pp. 142–150.

[100] Y. Su, N. Dupin, S. N. Parragh, and J. Puchinger, "A branch-and-price algorithm for the electric autonomous dial-a-ride problem," *Transportation Research Part B: Methodological*, vol. 186, p. 103011, 2024.

[101] V. Stallhofer and S. N. Parragh, "Event-based models for the electric autonomous dial-a-ride problem," *Transportation Research Part C: Emerging Technologies*, vol. 171, p. 104896, 2025.

[102] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 54, no. 1, pp. 7–22, 1991.

[103] L. B. Reinhardt, T. Clausen, and D. Pisinger, "Synchronized dial-a-ride transportation of disabled passengers at airports," *European Journal of Operational Research*, vol. 225, no. 1, pp. 106–117, 2013.

[104] S. Ropke, J. F. Cordeau, and G. Laporte, "Models and branch-and-cut algorithms for pickup and delivery problems with time windows," *Networks: An International Journal*, vol. 49, no. 4, pp. 258–272, 2007.

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

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

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

[108] L. Costa, C. Contardo, and G. Desaulniers, "Exact branch-price-and-cut algorithms for vehicle routing," *Transportation Science*, vol. 53, pp. 946–985, 2019.

[109] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks*, vol. 44, no. 3, pp. 216–229, 2004.

[110] M. Dror, "Note on the complexity of the shortest path models for column generation in vrptw," *Operations Research*, vol. 42, no. 5, pp. 977–978, 1994.

[111] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger, "Subset-row inequalities applied to the vehicle-routing problem with time windows," *Operations Research*, vol. 56, no. 2, pp. 497–511, 2008.

[112] I. Malheiros, R. Ramalho, B. Passeti, T. Bulhoes, and A. Subramanian, "A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem," *Computers & Operations Research*, vol. 129, p. 105196, 2021.

[113] S. Sohrabi, K. Ziarati, and M. Keshtkaran, "A hybrid genetic algorithm with an adaptive diversity control technique for the homogeneous and heterogeneous dial-a-ride problem," *Annals of Operations Research*, pp. 1–35, 2024.

[114] M. Karimi, F. Camiat, G. Desaulniers, and M. Gendreau, "An exact branch-and-price-and-cut algorithm for a practical and large-scale dial-a-ride problem," *Journal of the Operational Research Society*, vol. 76, no. 6, pp. 1125–1139, 2025.

[115] M. Diana and M. M. Dessouky, "A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows," *Transportation Research Part B: Methodological*, vol. 38, no. 6, pp. 539–557, 2004.

[116] Y. M. Nie and M. Ghamami, "A corridor-centric approach to planning electric vehicle charging infrastructure," *Transportation Research Part B: Methodological*, vol. 57, pp. 172–190, 2013.

[117] F. He, Y. Yin, and J. Zhou, "Deploying public charging stations for electric vehicles on urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 227–240, 2015.

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

[119] N. D. Kullman, J. C. Goodson, and J. E. Mendoza, "Electric vehicle routing with public charging stations," *Transportation Science*, vol. 55, no. 3, pp. 637–659, 2021.

APPENDIX A HETEROGENEOUS DIAL-A-RIDE FORMULATION

The heterogeneous dial-a-ride problem formulation used throughout the proposed VNS algorithm is reported in detail in this Appendix. This model serves as a common foundation for several components of our solution methodology. In particular, it is employed within the LP-based heuristic for generating initial solutions, provides the basis for the IP formulations used to define neighborhood classes in the VNS framework, and is also leveraged in the IP-based local search procedure. We extend a heterogeneous dial-a-ride problem in which both passengers and vehicles are of different types. Let $U = \{1, \ldots, n\}$ be the set of users (requests). For each $u \in U$, let the pickup node be u and the delivery node be $n + u$. Let $P = \{1, \ldots, n\}$, $D = \{n + 1, \ldots, 2n\}$, and $N = P \cup D \cup \{0, 2n + 1\}$ where 0 and $2n + 1$ are the origin and destination depots. Let K be the set of vehicles, and let M be the set of passenger types. Each request u has a type $m(u) \in M$; both nodes u and $n + u$ inherit this type. The travel time and the cost on the arc (i, j) for vehicle k are t^k_{ij} and c^k_{ij}, respectively. Node $i \in N$ has service time s^k_i and time window $[e_i, \ell_i]$. Vehicle k has route-duration limit T_k and per-type capacities Q^m_k for all $m \in M$. Let L_u be the maximum ride time for user u. Define the per-type node load $q^m_i > 0$ if $i = u \in P$ with $m(u) \in M$, $q^m_i < 0$ if $i = n + u \in D$ with $m(u) \in M$, and $q^m_i = 0$ otherwise. Let $\alpha_{k,m} \in \{0, 1\}$ indicate whether vehicle k can carry type m, and define $a_{k,i} = \alpha_{k,m(u)}$ for $i \in \{u, n + u\}$ and $a_{k,0} = a_{k,2n+1} = 1$. We use Big-M constants $M^k_{ij} \geq \max\{0,\ \ell_i + s^k_i + t^k_{ij} - e_j\}$ and $W^k_{ij,m} \geq \min\{Q^m_k,\ Q^m_k + q^m_i\}$.

Decision variables:

$x^k_{ij} \in \{0, 1\}$ equals 1 if vehicle k traverses (i, j);

$B^k_i \geq 0$ is the service start time of k at node i;

$L^k_u \geq 0$ is the ride time of user u on k;

$Q^k_{i,m} \geq 0$ is the number of type-m passengers on k *after* visiting i.

$$\min \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ij}^k\, x_{ij}^k \tag{A.1}$$

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1 \quad \forall i \in P \tag{A.2}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{n+i,j}^k = 0 \quad \forall i \in P,\ \forall k \in K \tag{A.3}$$

$$\sum_{j\in N} x_{0j}^k = 1 \quad \forall k \in K \tag{A.4}$$

$$\sum_{i\in N} x_{i,\,2n+1}^k = 1 \quad \forall k \in K \tag{A.5}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = 0 \quad \forall i \in P\cup D,\ \forall k \in K \tag{A.6}$$

$$B_j^k \geq B_i^k + s_i^k + t_{ij}^k - M_{ij}^k(1 - x_{ij}^k) \quad \forall i,j \in N,\ \forall k \in K \tag{A.7}$$

$$e_i \ \leq\ B_i^k \ \leq\ \ell_i \quad \forall i \in N,\ \forall k \in K \tag{A.8}$$

$$L_u^k = B_{n+u}^k - (B_u^k + d_u) \quad \forall u \in U,\ \forall k \in K \tag{A.9}$$

$$t_{u,\,n+u}^k \ \leq\ L_u^k \ \leq\ L_u \quad \forall u \in U,\ \forall k \in K \tag{A.10}$$

$$B_{2n+1}^k - B_0^k \leq T_k \quad \forall k \in K \tag{A.11}$$

$$Q_{j,m}^k \geq Q_{i,m}^k + q_i^m - W_{ij,m}^k(1 - x_{ij}^k) \quad \forall i,j \in N,\ \forall k \in K,\ \forall m \in M \tag{A.12}$$

$$\max\{0, q_i^m\} \ \leq\ Q_{i,m}^k \ \leq\ \min\{Q_k^m,\, Q_k^m + q_i^m\} \quad \forall i \in N,\ \forall k \in K,\ \forall m \in M \tag{A.13}$$

$$Q_{0,m}^k \ =\ Q_{2n+1,m}^k \ =\ 0 \quad \forall k \in K,\ \forall m \in M \tag{A.14}$$

$$x_{ij}^k \leq a_{k,i}, \qquad x_{ij}^k \leq a_{k,j} \quad \forall i,j \in N,\ \forall k \in K \tag{A.15}$$

$$x_{ij}^k \in \{0,1\} \quad \forall i,j \in N,\ \forall k \in K \tag{A.16}$$

$$B_i^k \geq 0 \quad \forall i \in N,\ \forall k \in K \tag{A.17}$$

$$L_u^k \geq 0 \quad \forall u \in U,\ \forall k \in K \tag{A.18}$$

$$Q_{i,M}^k \geq 0 \quad \forall i \in N,\ \forall k \in K,\ \forall M \in M. \tag{A.19}$$

The objective function (A.1) minimizes the total routing cost across all vehicles. Constraints (A.2) guarantee that each pickup node is visited exactly once across all vehicles, ensuring that every request is served. Constraints (A.3) enforce consistency between the pickup and delivery of each request: if vehicle k serves a pickup node i, then the same vehicle must also serve the corresponding delivery node $n+i$. Together, these constraints guarantee that each user is picked up and delivered exactly once by the same vehicle. Constraints (A.4)–(A.6) ensure that valid vehicle routes are formed. Constraints (A.4) force each vehicle k to leave its origin depot exactly once, while constraints (A.5) require each vehicle to arrive at its

destination depot exactly once. Constraints (A.6) maintain flow conservation at pickup and delivery nodes, meaning that if a vehicle enters a node, it must also leave it. These three sets of constraints together guarantee that each vehicle's route starts at the origin depot, serves a sequence of requests, and ends at the destination depot. Constraints (A.7) define the temporal relationship between consecutive nodes in a route: if vehicle k travels directly from node i to node j, then the service at j must begin no earlier than the completion of service at i plus the travel time. The Big-M linearization ensures that this condition is only active when arc (i, j) is used. Constraints (A.8) enforce that service at each node begins within its allowable time window $[e_i, \ell_i]$, thereby ensuring service feasibility. Constraints (A.9) define the ride time of each user as the difference between the service start at the delivery node and the departure from the pickup node. Constraints (A.10) bound this ride time between the direct travel time and the maximum allowable ride time L_u. This prevents solutions where users are transported excessively long compared to their direct journey. Constraints (A.11) limit the total duration of each vehicle's route, ensuring that it does not exceed the route duration limit T_k. Constraints (A.12)–(A.14) manage the type-specific capacities of vehicles. Constraints (A.12) propagate the load of type-m passengers carried by vehicle k along its route: after visiting node i, the load at the next node j must equal the load at i plus the change in demand q_i^m (positive for pickups, negative for deliveries). The Big-M form ensures that this update only applies when the arc (i, j) is traversed. Constraints (A.13) bound the onboard load of each passenger type between zero (or the minimum pickup) and the vehicle's capacity for that type. Constraints (A.14) initialize and terminate each route with zero load, ensuring vehicles leave and return empty. Constraints (A.15) enforce compatibility between passengers and vehicles. A vehicle k may only travel to a node i if it is capable of serving that passenger type, i.e., if $\alpha_{k,m(i)} = 1$. This prevents infeasible assignments, such as a standard vehicle being routed to pick up a wheelchair passenger when it lacks the required equipment. Finally, constraints (A.16) - (A.19) specify the domains of the decision variables.