HEC MONTRÉAL
École affiliée à l'Université de Montréal

Heuristic and Exact Methods for Workforce Scheduling and Routing Problems

par
Nicolás Cabrera Malik

Thèse présentée en vue de l'obtention du grade de Ph. D. en administration
(Spécialisation Gestion des opérations et de la logistique)

Mai 2025

HEC MONTRÉAL
École affiliée à l'Université de Montréal

Cette thèse intitulée :

Heuristic and Exact Methods for Workforce Scheduling and Routing Problems

Présentée par :

Nicolás Cabrera Malik

a été évaluée par un jury composé des personnes suivantes :

Yossiri Adulyasak
HEC Montréal
Président-rapporteur

Jean-François Cordeau
HEC Montréal
Codirecteur de recherche

Jorge E. Mendoza
HEC Montréal
Codirecteur de recherche

Marilène Cherkesly
Université du Québec à Montréal
Membre du jury

Dominique Feillet
École Nationale Supérieure des Mines de Saint-Étienne
Examinateur externe

Okan Arslan
HEC Montréal
Représentant du directeur de HEC Montréal

Résumé

Cette thèse explore le problème de planification et de tournées du personnel (WSRP), une variante du problème de tournées de véhicules qui vise à organiser les déplacements d'un groupe de travailleurs pour réaliser des tâches auprès d'une clientèle variée. Le WSRP est crucial dans des secteurs comme la maintenance du réseau électrique et les soins à domicile, où l'on cherche à optimiser l'efficacité, les coûts et la satisfaction des clients. Les approches classiques supposent souvent que les travailleurs peuvent se garer directement chez les clients, ce qui est peu réaliste en milieu urbain dense où le stationnement est limité. Cela peut causer des inefficacités, car les travailleurs doivent trouver une place de stationnement et se rendre à pied chez les clients, engendrant ainsi des retards et des surcoûts.

Cette thèse comble cette lacune en introduisant des itinéraires de type stationnement et boucle. Un tel itinéraire combine un trajet principal en véhicule avec des sous-tours effectués à pied ou en utilisant un mode de transport alternatif (par exemple, un scooter électrique ou un vélo). Dans le deuxième chapitre, nous définissons le problème des tournées avec stationnement et boucle et proposons une approche par décomposition et évaluation, reposant sur la génération de colonnes et de coupes, ainsi qu'un algorithme spécialisé pour le sous-problème. Notre méthode améliore la performance des métaheuristiques de pointe en termes de la qualité de la solution tout en restant compétitive en termes de temps de calcul.

Dans le troisième chapitre, nous abordons le WSRP avec stationnement et boucle pour une main-d'œuvre hétérogène, chaque travailleur ayant des compétences spé-

cifiques. Nous proposons une formulation basée sur la programmation linéaire en nombres entiers mixtes et nous développons un algorithme de séparation et évaluation basé sur le modèle du chapitre 2. Nos expériences montrent que cet algorithme améliore les performances des algorithmes de référence et peut aussi s'appliquer au problème de tournées et de planification des techniciens.

Enfin, dans le quatrième chapitre, nous étudions le cas où certaines demandes de clients apparaissent dynamiquement au cours de la journée. Nous développons des politiques de planification basées sur une heuristique d'échantillonnage multi-espace, qui génère des routes de haute qualité à chaque nouvelle demande, puis sélectionne la meilleure solution avec un modèle de partitionnement en ensembles. Nous proposons également une méthode pour calculer une borne sur le nombre de demandes pouvant être satisfaites sans hypothèse d'information complète.

Mots-clés

Routage, stationnement et boucle, heuristiques, partitionnement en ensembles, inégalités valides, séparation et évaluation avec génération de colonnes et coupes, algorithme de pulsation, programmation mathématique.

Méthodes de recherche

Recherche opérationelle, programmation mathématique, heuristiques

Abstract

This thesis addresses the workforce scheduling and routing problem (WSRP), a variant of the vehicle routing problem that involves scheduling and routing a set of workers to complete tasks for a diverse set of customers. The WSRP is prevalent in numerous fields, such as utility field operations and home healthcare, where companies seek to balance efficiency, cost, and customer convenience. Traditional approaches to the WSRP assume workers can park directly at customer locations, a simplification that is often unrealistic in dense urban areas with limited parking availability. This assumption can lead to inefficiencies, as workers may need to search for parking and walk to customer locations, resulting in delays and increased costs. In the first chapter, we present a comprehensive review of WSRPs and identify key open research areas.

This thesis contributes to filling the gap by studying the design of park-and-loop routes. A park-and-loop route is composed of a main tour that is carried out using a vehicle, and subtours that are performed on foot or by alternative modes of transport such as an electric scooter. In the second chapter, we introduce the park-and-loop routing problem, which consists in finding a set of least cost park-and-loop routes to serve a set of customers. We formulate the problem as a set covering problem and we propose an exact branch-price-and-cut method that relies on the pulse algorithm to solve the pricing problem. We also present a set of acceleration strategies that boost performance. The algorithm outperforms three state-of-the-art metaheuristics in terms of solution quality and compares reasonably in terms of the computational

effort.

In the third chapter, we study another practical and difficult problem known as the workforce scheduling and routing problem with park-and-loop. As opposed to the problem described in Chapter 2, the set of workers is not homogeneous. That is, each worker in the workforce masters a subset of skills with a potentially different level of proficiency. Similarly, customer requests may require one or more skills, each at a given level of proficiency. We first formulate the problem as a mixed integer program model with a polynomial number of constraints and variables, and then we present a path-based formulation that has an exponential number of variables. To solve the latter, we propose a branch-price-and-cut algorithm that builds on the method described in Chapter 2. Our experiments show that our algorithm outperforms the benchmark algorithms.

In the fourth chapter, we extend the scope to consider the case where some of the customer demand is revealed dynamically throughout the workday. To address this problem, we propose a set of scheduling policies that build on the multi-space sampling heuristic. Every time a new customer request is revealed, this method generates a set of high-quality routes and then it builds a new solution by selecting from these routes using a set partitioning model. We also present a method to compute a bound on the number of customer requests that can be completed under the assumption of complete information.

Keywords

Routing, park-and-loop, heuristics, set partitioning, valid inequalities, branch-price-and-cut, pulse algorithm, mathematical programming

Research Methods

Operations research, mathematical programming; heuristics

Contents

List of Tables

List of Figures

List of Acronyms

ALNS	Adaptive Large Neighborhood Search
BFS	Best-first Search
BPC	Branch-price-and-cut
CDPP	Capacitated Delivery Problem with Parking
CG	Column Generation
CSP	Constrained Shortest Path
DFS	Depth-first Search
DOPLRP	Doubly Open Park-and-loop Routing Problem
ESPPRC	Elementary Shortest Path Problem with Resource Constraints
ESPPRC-PL	Elementary Shortest Path Problem with Resource Constraints and Park-and-loop
2ELRP	Two-echelon Location Routing Problem
2E-LMDP	Two-echelon Last-mile Delivery Problem
2EVRP	Two-echelon Vehicle Routing Problem
2EVRP-TM	Two-echelon Vehicle Routing Problem with Time Windows and Mobile Satellites

ILS	Iterated Local Search
MILP	Mixed Integer Linear Program
MIP	Mixed Integer Program
MSH	Multi-space Sampling Heuristic
PA	Pulse Algorithm
PLRP	Park-and-loop Routing Problem
PLRP-PS	Park-and-loop Routing Problem with Parking Selection
RSCP	Relaxed Set Covering Problem
SC	Set Covering
SLNS	Small and Large Neighborhood Search
STTRPSD	Single Truck and Trailer Routing Problem with Satellite Depots
TRSP	Technician Routing and Scheduling Problem
TSP	Traveling Salesman Problem
TTRP	Truck and Trailer Routing Problem
TTRPTW	Truck and Trailer Routing Problem with Time Windows
WSRP	Workforce Scheduling and Routing Problem
WSRP-PL	Workforce Scheduling and Routing Problem with Park-and-loop
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPD	Vehicle Routing Problem with Drones
VRPTR	Vehicle Routing Problem with Transportable Resources

To

Maria, Martin, Oscar, Sandra, Sebastian,

& Thor

Acknowledgements

This thesis would not have come to fruition without the invaluable support of many people along the way. First and foremost, I am deeply grateful to my advisors, Jorge and Jean-François, for their guidance and patience throughout my years at HEC Montréal and my internship at CIRRELT. Their dedication to my development as a researcher has been instrumental in shaping this work. I am also sincerely grateful to them for encouraging and supporting my attendance to international conferences, where I had the opportunity to present our research.

I also extend my heartfelt thanks to my CIRRELT colleagues, whose support has been invaluable over the past four years. Working alongside such talented and dedicated individuals made this experience not only productive but truly enjoyable. I am especially grateful to Jordi for his friendship, the lively discussions, and the countless moments of laughter that brightened even the most challenging days. Thank you for always taking the time to review the first drafts of my articles and for offering constructive feedback on all the key presentations I gave throughout my PhD. Your insights and encouragement have greatly enriched this journey.

I also want to express my deepest gratitude to my family and friends, whose support has been key. To my family, thank you for your constant encouragement, patience, and understanding during times of stress. To my dad and my brothers, I thank you for pushing my limits. I had tons of fun training for all the marathons we did in the last four years. To my friends, Jitsama, Tapia, Judith, Sebas, François, Tommaso, Annie, and Carolina. Thank you for being there to lift my spirits, cel-

ebrate small victories, and remind me of life beyond research. I am incredibly fortunate to have you all.

Lastly, I thank the jury for their thorough reading of this document.

General Introduction

The workforce scheduling and routing problem (WSRP) is one of the most widely studied variants of the vehicle routing problem. The WSRP involves the transportation and scheduling of a set of workers (e.g., technicians, nurses, and security guards) to fulfill different tasks for a set of heterogeneous customers. This problem naturally arises in a wide range of applications. For example, Cabrera et al. (2022) consider the case of a large electricity maintenance provider in France where technicians perform meter readings at customer locations. Similarly, Zamorano and Stolletz (2017) consider the case of an external maintenance service provider specialized in the repair and maintenance of electric forklifts. In the field of homecare, Braekers et al. (2016) consider the case of an organization in Austria that schedules nurses and care workers to visit patients while taking into account the trade-off between the cost and the client inconvenience. Other applications include the scheduling of personnel at consolidation and transshipment ports (Li et al. 2005) and security personnel performing routine rounds (Misir et al. 2011).

The standard assumption in the WSRP literature is that workers can and should always park their vehicles at customer locations, thus following *pure vehicle routes*. Although this assumption can facilitate the design of the routing plan for each worker, it can be wildly optimistic in practice, particularly, when customers are located in urban areas such as city centers. Indeed, these locations are often subject to restricted mobility and scarce access to parking. As a result, workers may not be able to park directly at every customer (as suggested by the routing plan) and

instead may be obligated to cruise for parking. In that case, only after finding a suitable parking spot, they can walk to serve the customer. Cruising for parking may take up to five minutes in a city center (Reed et al. 2024) and can significantly increase the total duration of a route. This can be troubling as workers are often subject to tight schedules. As a result, companies end up paying millions of dollars in delays and overtime fees (Chiara et al. 2020).

Furthermore, when routing plans do not consider parking decisions, workers are often faced with the task of modifying the routes on the fly. Indeed, workers have to choose between visiting other nearby customers or returning to the parking spot to recover the vehicle. Although workers' experience may help them to make a decision, it may not be the correct one. Le Colleter et al. (2023) found that even when parking times are low, walking to serve several customers departing from the same parking spot can reduce driving time by more than 15% and parking times by more than 50%. Similarly, Cabrera et al. (2022) showed that costs can be reduced up to 12% as a result of better balancing the use of driving and walking, and by decreasing the number of workers needed to visit all the customers. Thus, significant productivity gains can be expected by optimizing parking decisions.

In this thesis, we investigate the design of efficient *park-and-loop* routes. Formally, a park-and-loop route consists of a main tour that is completed using a vehicle (i.e., a car, a van, a truck) and subtours, visiting one or more customers, that are carried out after safely parking the vehicle, using a more flexible transportation mode (i.e., on foot, by bike, by electric scooter). Furthermore, we define three new variants of the WSRP. First, we introduce the park-and-loop routing problem (PLRP). This problem consists of designing minimum-cost park-and-loop routes for a set of workers while ensuring a maximum duration for each subtour and a maximum total walking distance. Second, we describe the workforce scheduling and routing problem with park-and-loop (WSRP-PL). This problem generalizes the PLRP by taking into account the skills compatibility between workers and tasks, and the presence of time windows at customer locations. Third, we introduce the dynamic park-and-

loop routing problem (DPLRP). This problem extends the PLRP by considering both scheduled and on-demand requests. Scheduled requests are known beforehand and as such must be completed before the end of the planning horizon. On the contrary, on-demand requests appear randomly throughout the day, and it is possible to reject them.

This thesis is composed of four chapters. Each chapter is a self-contained paper.

Chapter 1 presents:

Cabrera, N., Cordeau, J.-F, & Mendoza, J.E. Workforce Scheduling and Routing Problems: A Literature Review. Working-paper.

In this paper, we present a comprehensive review of WSRPs. To do so, we start by introducing two mathematical formulations for the problem and thoroughly discuss their main advantages and disadvantages. Then, we describe the most relevant features addressed in the literature and classify each article accordingly. By presenting these features we identify key open research areas on the WSRP. In particular, we emphasize the importance of developing new methodologies to solve WSRPs that include multi-modal routes and precedence constraints. In addition, we present an overview of the solution methods and we describe the most common applications of the WSRP. Finally, we present and discuss multiple practical considerations to build more applicable and successful solutions.

Chapter 2 presents:

Cabrera, N., Cordeau, J.-F, & Mendoza, J.E. (2023) Solving the Park-and-loop Routing Problem by Branch-price-and-cut. Transportation Research Part C: Emerging Technologies, 107, p. 104369.

In this paper, we present the PLRP in detail. Moreover, we describe an exact method capable of optimally solving 39 out of 40 instances with up to 50 customers belonging to what is probably the most studied testbed in the PLRP literature. The algorithm is based on the branch-price-and-cut framework. To solve the pricing problem we extend the pulse algorithm to generate park-and-loop routes and propose a new pruning strategy that significantly improves the algorithm's performance. We

rigorously evaluate the impact of each component of the proposed algorithm. We also present a detailed comparison with three state-of-the-art metaheuristics. Finally, we evaluate the impact of considering park-and-loop routes on the total driving distance and the final solution structure. Our experiments show that introducing park-and-loop routes can decrease the driven distance by up to 18%.

Chapter 3 presents:

Cabrera, N., Cordeau, J.-F, & Mendoza, J.E. (2025) The Workforce Scheduling and Routing Problem with Park-and-loop. Networks, 85, pp. 38-60.

In this paper, we introduce the WSRP-PL and two mixed integer programming models to solve it. The first model is an arc-based formulation with a polynomial number of constraints and variables concerning the number of customers and workers. However, the bound obtained by solving its LP relaxation is weak. The second model is a path-based formulation that has an exponential number of variables. To solve the latter, we propose a branch-price-and-cut algorithm that builds on the method described in Chapter 2. A key component of this algorithm is the pricing problem solver. The solver simultaneously builds a team of workers and the route they follow. Moreover, the solver evaluates multiple team compositions in parallel, which significantly speeds up the algorithm. We generate a large testbed consisting of small to large instances and compare the performance of a mixed integer programming solver that solves the arc-based formulation (i.e., CPLEX) with the proposed algorithm. Our experiments show that our algorithm is faster and can optimally solve more instances. Finally, we test the performance of our algorithm on a related problem for which the algorithm finds 12 new best solutions.

Chapter 4 presents:

Cabrera, N., Cordeau, J.-F, & Mendoza, J.E. The Dynamic Park-and-Loop Routing Problem. Working-paper.

In this paper, we introduce the dynamic park-and-loop routing problem, an extension of the park-and-loop routing problem in which some of the customer demand is revealed dynamically throughout the workday. Each time a new customer request is

received, the current plan must be re-evaluated and, in many cases, revised to accommodate the new request. The objective is to maximize the number of requests served. To address this problem, we propose a set of myopic and anticipatory scheduling policies that build on a very fast metaheuristic known as the multi-space sampling heuristic, a two-phase approach that follows the route-first, assemble-second principle. Every time a new customer request is revealed, this method generates a set of high-quality routes and then it builds a new solution by selecting from these routes using a set partitioning model. We evaluate all policies on a new set of instances with up to 100 requests, considering various spatio-temporal demand distributions. We also present a method to compute a bound on the number of customer requests that can be completed under the assumption of complete information.

Finally, we summarize the main contributions of this thesis and point to several potential future research avenues.

References

Braekers, K., R. F. Hartl, S. N. Parragh, and F. Tricoire (2016). "A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience". *European Journal of Operational Research* 248.2, pp. 428–443.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2022). "The doubly open park-and-loop routing problem". *Computers & Operations Research* 143, p. 105761.

Chiara, G. D., L. Cheah, C. L. Azevedo, and M. E. Ben-Akiva (2020). "A policy-sensitive model of parking choice for commercial vehicles in urban areas". *Transportation Science* 54.3, pp. 606–630.

Le Colleter, T., D. Dumez, F. Lehuédé, and O. Péton (2023). "Small and large neighborhood search for the park-and-loop routing problem with parking selection". *European Journal of Operational Research* 308.3, pp. 1233–1248.

Li, Y., A. Lim, and B. Rodrigues (2005). "Manpower allocation with time windows and job-teaming constraints". *Naval Research Logistics (NRL)* 52.4, pp. 302–311.

Misir, M., P. Smet, K. Verbeeck, and G. Vanden Berghe (2011). "Security personnel routing and rostering: a hyper-heuristic approach". *Proceedings of the 3rd International Conference on Applied Operational Research*. Vol. 3. Tadbir; Canada, pp. 193–205.

Reed, S., A. M. Campbell, and B. W. Thomas (2024). "Does parking matter? The impact of parking time on last-mile delivery optimization". *Transportation Research Part E: Logistics and Transportation Review* 181, p. 103391.

Zamorano, E. and R. Stolletz (2017). "Branch-and-price approaches for the multi-period technician routing and scheduling problem". *European Journal of Operational Research* 257.1, pp. 55–68.

Chapter 1

Workforce Scheduling and Routing Problems: A Literature Review

Abstract

The workforce scheduling and routing problem (WSRP) involves assigning geographically dispersed tasks to workers and planning their routes to complete these tasks efficiently. This problem arises in numerous real-world scenarios, including technicians conducting preventive maintenance at customer sites, nurses providing home care, and security guards patrolling multiple locations. To address these challenges, researchers have incorporated a wide range of constraints, such as time windows, skill compatibility, and team building. In this study, we systematically structure and analyze the WSRP literature, identifying its core characteristics and uncovering key research gaps. Our findings highlight critical areas for future investigation, including the integration of multi-modal routes and precedence constraints. Additionally, we emphasize practical features that should guide the development of new solution methods for this family of problems, ensuring their applicability to real-world workforce management challenges.

1.1 Introduction

Vehicle routing problems (VRPs) have been a central focus for both academics and practitioners for decades, driven by their widespread real-world applications. Businesses and organizations rely on VRP solutions in diverse fields, including distribution logistics, laundry services, and urban waste management. Researchers worldwide have dedicated significant efforts to modeling and solving countless VRP variants, striving to bridge the gap between theory and practice. For a comprehensive overview of existing and emerging VRP variants, we refer the reader to Vidal et al. (2020). In this paper, we focus on VRP variants that arise in service-oriented contexts. Specifically, we examine problems that involve scheduling and routing workers to perform tasks at multiple locations—collectively referred to as workforce scheduling and routing problems (WSRPs).

WSRPs combine elements from both scheduling and routing problems, both of which are NP-hard (Goel and Meisel 2013). The scheduling aspect involves assigning workers—such as technicians, nurses, and security guards—to perform services or complete tasks for customers. Worker assignments must account for various constraints, including skill compatibility (when only certain workers are qualified for a task), team building (when multiple workers are required), service priorities, and task precedence. The routing aspect focuses on designing efficient travel routes for workers moving between locations. Several critical factors influence route planning, including time windows, the presence of one or multiple depots, and the availability of different transportation modes. This review concentrates on problems that integrate both scheduling and routing. Note that we exclude in our review pick-up and delivery problems because no significant work occurs at the customer location. We also do not consider studies that address only the scheduling component and instead refer the interested reader to Estellon et al. (2009), Cordeau et al. (2010), Hashimoto et al. (2011), and Fırat and Hurkens (2012) for detailed discussions on pure scheduling problems.

WSRPs naturally emerge in a wide range of real-world applications. Goel and Meisel (2013) examined a large electricity network in central Germany, where technicians perform maintenance on power infrastructure. Similarly, Çakırgil et al. (2020) tackled a scheduling problem for an energy distribution company in Turkey, optimizing technician assignments for failure diagnosis and corrective maintenance. In home health care, Algethami et al. (2019) studied six UK-based scenarios where nurses and caregivers are scheduled to visit patients at home. Other notable applications include personnel scheduling at airport check-in counters (Zamorano et al. 2018) and security officers conducting routine patrols (Misir et al. 2011).

The widespread applicability of WSRPs has drawn significant interest from researchers across various fields. Depending on the application, different studies have introduced distinct names and acronyms for their specific problem variants, such as the technician scheduling and routing problem (TSRP), the health care scheduling and routing problem (HSRP), and the manpower allocation and routing problem (MARP), among others. This lack of uniformity has led to two major challenges. First, the field lacks a clear consensus on the state-of-the-art algorithms or methodologies for solving WSRPs. Second, there is no standardized terminology for distinguishing between different problem variants. The most recent review on WSRPs, conducted by Paraskevopoulos et al. (2017), addresses these issues by proposing a taxonomy based on three key categories: resource qualifications, service requirements, and objectives. Their work also provides an overview of the main solution methodologies and suggests directions for future research. However, the field has evolved significantly since then, with more than 30 new studies published after their review.

This paper provides an up-to-date, comprehensive literature review of WSRPs, covering key constraints and datasets. As a foundational step for advancing the field, this review equips researchers with the necessary insights to develop new solution approaches and identify critical directions for future research. Our contribution is twofold. First, we synthesize the evolution of the field—past, present,

and future—by systematically analyzing the existing literature. Second, unlike Paraskevopoulos et al. (2017), we emphasize methodological advancements and practical considerations, offering guidance that will help researchers design more applicable and effective solutions.

The remainder of this article is structured as follows. Section 1.2 formally defines the baseline WSRP and presents two mathematical formulations of the problem. Section 1.3 reviews the key features explored in the literature, while Section 1.4 highlights emerging practical aspects that remain largely unaddressed. Finally, Section 1.5 concludes the paper and outlines directions for future research.

1.2 Problem description

In this section, we introduce a canonical WSRP, which serves as a fundamental framework for understanding and extending the problem in various directions. Many WSRP variants build upon this core structure by incorporating additional constraints, objectives, or problem-specific characteristics. We first provide a formal problem definition that captures the essential features of a WSRP. We then present two mixed integer programming formulations: an arc-based model and a route-based model. These formulations not only offer a formal mathematical representation of the problem, but also serve as a reference point for analyzing more complex extensions explored later in this review.

1.2.1 Problem statement

The WSRP can be formally defined on a complete graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of all nodes and $\mathcal{A}$ is the set of arcs. The set of nodes comprises a start depot $\underline{0}$, the set of tasks $\mathcal{C} = \{1, \ldots, n\}$, and an end depot $\overline{0}$. Arcs in $\mathcal{A}$ represent the connections between two nodes. Traveling along arc $(i, j) \in \mathcal{A}$ has a cost c_{ij} and a travel time t_{ij}. To complete tasks a set of workers $\mathcal{W}$ are assigned to teams in the set

$\mathcal{T}$. A maximum of Γ workers can be assigned to a team. Worker qualifications are represented by p_{wsl}, which is a binary parameter equal to 1 if worker $w \in \mathcal{W}$ has at least a proficiency level $l \in \mathcal{L}$ for skill $s \in \mathcal{S}$. On the other hand, skill requirements are represented by q_{isl}, a binary parameter stating if task $i \in \mathcal{C}$ needs a worker with at least a proficiency level $l \in \mathcal{L}$ for skill $s \in \mathcal{S}$.

Each task $i \in \mathcal{C}$ has a duration d_i and an associated time window indicating possible visit times. Let $[a_i, b_i]$ be the earliest and latest starting time of task $i \in \mathcal{C}$. All teams depart from the start depot at time a_0 and must return before time b_0. If the assembled teams cannot fulfill a certain task, it may be outsourced at a cost o_i. The objective of the WSRP is to minimize the total cost while ensuring that each task is fulfilled precisely once and the total duration of each route does not exceed the working day duration.

1.2.2 Arc-based formulation

In the following, we provide a model similar to the one in Kovacs et al. (2012) and Xie et al. (2017). The model uses the following binary variables:

- $x_{ij}^t = 1$ if team $t \in \mathcal{T}$ travels from node i to node j, for $(i, j) \in \mathcal{A}$ and 0, otherwise,
- $y_i^t = 1$ if team $t \in \mathcal{T}$ is assigned to complete task $i \in \mathcal{C}$ and 0, otherwise,
- $v_w^t = 1$ if worker $w \in \mathcal{W}$ is assigned to team $t \in \mathcal{T}$ and 0, otherwise,
- $h_t = 1$ if team $t \in \mathcal{T}$ is selected and 0, otherwise,
- $z_i = 1$ if task $i \in \mathcal{C}$ is outsourced and 0, otherwise.

We also define the following continuous variables:

- u_i^t if the arrival time of team $t \in \mathcal{T}$ to node $i \in \mathcal{N}$.

The arc-based formulation (ABF) for the WSRP is stated as follows:

$$\min \sum_{(i,j)\in\mathcal{A}} \sum_{t\in\mathcal{T}} c_{ij} x_{ij}^t + \sum_{i\in\mathcal{C}} o_i z_i \tag{1.1}$$

subject to

$$\sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{N}} x_{ij}^t + z_i = 1 \qquad \forall i \in \mathcal{C} \tag{1.2}$$

$$\sum_{j\in\mathcal{N}} x_{\bar{0}j}^t = 1 \qquad \forall t \in \mathcal{T} \tag{1.3}$$

$$\sum_{i\in\mathcal{N}} x_{i\bar{0}}^t = 1 \qquad \forall t \in \mathcal{T} \tag{1.4}$$

$$\sum_{i\in\mathcal{N}} x_{ij}^t - \sum_{i\in\mathcal{N}} x_{ji}^t = 0 \qquad \forall j \in \mathcal{C}, t \in \mathcal{T} \tag{1.5}$$

$$(t_{ij} + d_i) x_{ij}^t + u_i^t \leq u_j^t + b_i (1 - x_{ij}^t) \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{1.6}$$

$$u_i^t \leq a_i \qquad \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{1.7}$$

$$u_i^t \geq b_i \qquad \forall i \in \mathcal{N}, t \in \mathcal{T} \tag{1.8}$$

$$\sum_{t\in\mathcal{T}} v_w^t \leq 1 \qquad \forall w \in \mathcal{W} \tag{1.9}$$

$$\sum_{w\in\mathcal{W}} p_{wsl} v_w^t \geq q_{isl} y_i^t \qquad \forall t \in \mathcal{T}, s \in \mathcal{S}, l \in \mathcal{L}, i \in \mathcal{C} \tag{1.10}$$

$$\sum_{j\in\mathcal{N}} x_{ji}^t = y_i^t \qquad \forall i \in \mathcal{C}, t \in \mathcal{T} \tag{1.11}$$

$$\sum_{w\in\mathcal{W}} v_w^t \geq h_t \qquad \forall t \in \mathcal{T} \tag{1.12}$$

$$\sum_{w\in\mathcal{W}} v_w^t \leq \Gamma h_t \qquad \forall t \in \mathcal{T} \tag{1.13}$$

$$h_t \geq h_{t+1} \qquad \forall t \in \mathcal{T} | t < |\mathcal{T}| \tag{1.14}$$

$$\sum_{i\in\mathcal{C}} y_i^t \geq h_t \qquad \forall t \in \mathcal{T} \tag{1.15}$$

$$x_{ij}^t \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{1.16}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{C} \tag{1.17}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in \mathcal{C}, t \in \mathcal{T} \tag{1.18}$$

$$v_w^t \in \{0,1\} \qquad \forall w \in \mathcal{W}, t \in \mathcal{T} \tag{1.19}$$

$$h_t \in \{0,1\} \qquad \forall t \in \mathcal{T} \tag{1.20}$$

$$u_i^t \geq 0 \qquad \forall i \in \mathcal{N}, t \in \mathcal{T}. \tag{1.21}$$

The objective function (1.1) minimizes the total cost comprised of the transportation and outsourcing costs. Constraints (1.2)-(1.8) represent the routing part of the problem. Constraints (1.2) ensure that all tasks are either completed or outsourced. Constraints (1.3) guarantee that all teams leave the start depot. Constraints (1.4) state that all the teams must arrive at the end depot. Constraints (1.5) are the flow balance constraints. Constraints (1.6) define the arrival time to every node. These constraints are subtour elimination constraints in the form of the Miller-Tucker-Zemlin inequalities. Constraints (1.7) and (1.8) enforce the time window restrictions. The remaining constraints, i.e., (1.9)-(1.15), are the scheduling constraints. Constraints (1.9) state that a worker can only be assigned to one team. Constraints (1.10) and (1.11) ensure that tasks are only completed by teams with the appropriate qualifications. Constraints (1.12) and (1.13) impose a minimum and maximum number of workers for each selected team. Constraints (1.14) remove symmetric solutions concerning the selection of the teams. Constraints (1.15) guarantee that all selected teams complete at least one task. Constraints (1.16)-(1.21) state the variable domains.

The ABF is compact, as it contains a polynomial number of variables and constraints. However, due to the large number of binary variables, the quality of the lower bound obtained by solving its LP relaxation is very poor. As a result, general-purpose solvers (i.e., Gurobi, CPLEX) can only solve small instances, with a few workers and customers.

1.2.3 Route-based formulation

The WSRP can also be modeled with a route-based formulation. Let $\mathcal{R}$ denote the set of feasible routes, where a route is an ordered set of arcs starting and ending at the depot. A route r is feasible if and only if it satisfies the time windows and the skills compatibility constraints. The parameter c_r is the total cost of using route r. Let g_{ir} be a binary parameter that takes the value of 1 if and only if route $r \in \mathcal{R}$ completes task $i \in \mathcal{C}$. Also, let m_{wr} be a binary parameter that takes the value of 1 if and only if route $r \in \mathcal{R}$ uses worker $w \in \mathcal{W}$. The binary variable λ_r is equal to one if and only if the route r is selected. Then, the route-based formulation (RBF) for the WSRP is stated as follows:

$$\min \sum_{r \in \mathcal{R}} c_r \lambda_r + \sum_{i \in \mathcal{C}} o_i z_i \tag{1.22}$$

subject to (1.17) and

$$\sum_{r \in \mathcal{R}} g_{ir} \lambda_r + z_i = 1 \qquad \forall i \in \mathcal{C} \tag{1.23}$$

$$\sum_{r \in \mathcal{R}} m_{wr} \lambda_r \leq 1 \qquad \forall w \in \mathcal{W} \tag{1.24}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in \mathcal{R}. \tag{1.25}$$

The objective function minimizes the total transportation and outsourcing costs. Constraints (1.23) ensure that all tasks are either completed by one team or outsourced. Constraints (1.24) state that only one route can be selected for each worker. Constraints (1.25) define the variables domain. As opposed to the ABF, the RBF has an exponential number of variables. Enumerating all these variables (i.e., routes) is only possible for trivial instances. As an alternative, route-based formulations are typically solved by using column generation (CG). The intuition behind CG is to find the optimal solution without explicitly considering all variables. To do so, two

optimization problems are solved sequentially: A *master problem* that considers only a small subset of variables and a *pricing problem* that delivers promising variables to the master problem. More details about CG approaches can be found in Uchoa et al. (2024) and in Desrosiers et al. (2024).

In the WSRP, the master problem corresponds to the relaxed version of formulation (1.22)-(1.25). The relaxed RBF considers only a subset of feasible routes $\overline{\mathcal{R}} \subseteq \mathcal{R}$ and is obtained by relaxing the binary constraints on the λ_r variables. The pricing problem is the route generator. Let $\pi_i \in \mathbb{R}$ and $\sigma_w \leq 0$ be the dual variables associated with constraints (1.23) and (1.24) respectively. Then, the pricing problem can be stated by dropping the team t index from the ABF, updating the objective function, and by slightly modifying constraints (1.2). More specifically, the pricing problem is stated as follows:

$$\min \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} - \sum_{i\in C} \pi_i y_i - \sum_{w\in\mathcal{W}} \sigma_w v_w \tag{1.26}$$

subject to

$$\sum_{j\in\mathcal{N}} x_{ij} = y_i \qquad \forall i \in C \tag{1.27}$$

$$\sum_{j\in\mathcal{N}} x_{\bar{0}j} = 1 \tag{1.28}$$

$$\sum_{i\in\mathcal{N}} x_{i\bar{0}} = 1 \tag{1.29}$$

$$\sum_{i\in\mathcal{N}} x_{ij} - \sum_{i\in\mathcal{N}} x_{ji} = 0 \qquad \forall j \in C \tag{1.30}$$

$$(t_{ij} + d_i)x_{ij} + u_i \leq u_j + b_i(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A} \tag{1.31}$$

$$u_i \leq a_i \qquad \forall i \in \mathcal{N} \tag{1.32}$$

$$u_i \geq b_i \qquad \forall i \in \mathcal{N} \tag{1.33}$$

$$\sum_{w\in\mathcal{W}} v_w \leq \Gamma \tag{1.34}$$

$$\sum_{w \in \mathcal{W}} p_{wsl} v_w \geq q_{isl} y_i \qquad \forall s \in \mathcal{S}, l \in \mathcal{L}, i \in C \tag{1.35}$$

$$\sum_{j \in \mathcal{N}} x_{ji} = y_i \qquad \forall i \in C \tag{1.36}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{1.37}$$

$$y_i \in \{0,1\} \qquad \forall i \in C \tag{1.38}$$

$$v_w \in \{0,1\} \qquad \forall w \in \mathcal{W} \tag{1.39}$$

$$u_i \geq 0 \qquad \forall i \in \mathcal{N}. \tag{1.40}$$

The objective function (1.26) aims to minimize the reduced cost of the route. Constraints (1.27)-(1.36) are equivalent to constraints (1.2)-(1.11). The pricing problem can also be modeled as an elementary shortest path problem with resource constraints (ESPPRC) as in Cabrera et al. (2023). To solve the ESPPRC one usually resorts to dynamic programming algorithms such as the labeling algorithm introduced by Feillet et al. (2004). For more details regarding the ESPPRC and solution methods, we refer the reader to Costa et al. (2019a).

1.3 The WSRP and its variants

Over the years, researchers have extended the basic WSRP to better reflect real-world applications. The following subsections outline key problem variations studied in the literature, providing concrete examples and discussing their implications. We also examine how each feature impacts problem complexity and solution methodologies.

1.3.1 Skill compatibility

The need for specific skills to complete tasks is a key feature in many WSRPs. When this constraint is included, only a subset of the workforce is qualified for certain tasks, and task durations may vary depending on the worker's skill level. This shift

introduces complexity, moving from a homogeneous (often unlimited) workforce to a heterogeneous (typically limited) one.

One of the earliest studies incorporating skill compatibility in WSRPs was by Tsang and Voudouris (1997), who examined British Telecom's workforce scheduling problem. In this case, service time depended on a worker's skill factor. For instance, if a worker had a skill factor of 0.9 and a task had a standard duration of 20 minutes, the worker would complete it in 18 minutes. Accounting for such variations is crucial for optimizing workforce schedules, as it enables better task allocation, reduces idle time, and can ultimately decrease the total number of workers needed. To tackle this problem, Tsang and Voudouris (1997) proposed a fast local search heuristic capable of producing solutions in approximately three hours. Over a decade later, Günther and Nissen (2013) improved solution quality using a particle swarm optimization algorithm, further minimizing idle time and lowering overall costs. However, this approach did not improve computational efficiency.

Beyond varying task durations, some WSRP applications impose stricter constraints, where only a subset of workers can perform certain tasks. For example, Li et al. (2005) studied workforce scheduling at the Port of Singapore, where a pool of heterogeneous workers must be dispatched from a central hub to complete tasks across the port. Proper worker-task assignments in this context is critical, as misassignments could disrupt operations — an aspect overlooked in previous research by Lim et al. (2004). Li et al. (2005) addressed this gap by introducing a simulated annealing heuristic, which improved prior solutions while incorporating skill constraints.

More recently, researchers explore additional aspects of skill compatibility. Braekers et al. (2016) study a home care organization in Austria, where nurses are scheduled to perform tasks at patient locations. Beyond skill constraints, patients have preferences for specific nurses, and not being attended by their preferred caregiver reduces their perceived level of service. To address this, the authors propose a metaheuristic that explicitly considers the trade-off between minimizing total costs and

reducing patient inconvenience. Their results show that minimal-cost solutions can be significantly improved in terms of reducing patient inconvenience with only a slight increase in overall costs, even for instances with up to 300 tasks.

Chen et al. (2016) examine how workers can acquire new skills through experience. They model a heterogeneous workforce where initial proficiency levels and learning rates are known. As workers complete tasks, their experience grows, allowing them to reduce service times or take on more complex assignments. Their study demonstrates that explicitly modeling skill acquisition leads to better and more adaptive solutions compared to approaches that assume static proficiency levels. Despite their potential, learning effects and patient preferences remain underexplored in the WSRP literature, highlighting a key area for future research.

1.3.2 Team building

Closely related to skill compatibility, team building is often necessary to complete tasks requiring multiple workers. This occurs when a task demands two or more equally skilled workers or when different skill sets must be combined.

Bredström and Rönnqvist (2008) study a home care scheduling problem where many patients require synchronized visits from multiple caregivers. The authors first propose a mixed-integer programming model, but as the number of tasks and workers increases, the model's complexity grows exponentially, making it impractical for large instances. To address this, they develop a heuristic method that finds solutions in under 10 minutes for instances with up to 60 tasks. This work highlights how synchronizing the schedules and routes of multiple workers significantly increases WSRP complexity.

In many applications, tasks require workers with complementary skills. Zamorano and Stolletz (2017) investigate an external maintenance service provider specializing in electric forklifts, where maintenance and failure diagnosis tasks demand multiple skills. To address this, the authors propose a branch-and-price approach capable of

solving instances with up to 60 tasks exactly. Their results emphasize the importance of problem decomposition to handle large-scale instances efficiently. Similarly, Anoshkina and Meisel (2019) explore a setting where teams must be formed while considering worker skills and experience levels. Unlike previous studies, they incorporate three simultaneous objectives: service quality, workload fairness, and cost minimization. To tackle this, they introduce a bi-level decomposition approach that separates team rostering from routing and scheduling, reinforcing the need for decomposition techniques in solving large-scale WSRPs.

Most studies assume that teams remain together throughout the planning horizon. However, in some scenarios, partial team splitting improves operational efficiency. Dohn et al. (2009) analyze ground handling operations at a major European airport, where workers handle baggage and cleaning tasks between aircraft arrivals and departures. In this setting, workers cooperate for specific tasks before continuing their schedules independently. The authors develop a branch-and-price algorithm that finds optimal solutions for instances with up to 150 tasks.

Similarly, Rasmussen et al. (2012) study a Danish healthcare center where home carers visit patients requiring either multiple caregivers or exclusive skill combinations. Their model allows partial team splitting, leading to more efficient solutions than those found in Bredström and Rönnqvist (2008), where teams remained fixed. However, this added flexibility increases problem complexity, and instances with over 100 tasks remain unsolved optimally. Other studies considering team splitting include Dohn et al. (2009), Kovacs et al. (2012), and Zhan and Wan (2018).

Future research should further explore decomposition approaches to enhance the scalability of WSRP solutions, particularly for large instances involving complex team structures.

1.3.3 Outsourcing

In many contexts, it is not always feasible to complete all tasks with the available workforce. Companies may therefore rely on outsourcing tasks to external providers or, in some cases, leave tasks unfulfilled while incurring a penalty cost. This feature increases the complexity of the WSRP, as decisions about whether to fulfill, defer, or outsource tasks must be integrated into the model. When outsourcing is an option, researchers must adapt both mathematical models and heuristic procedures. This often requires additional decision variables in exact models or modified insertion operators in heuristic methods to properly balance internal workforce scheduling with outsourcing costs.

Dohn and Sevel (2008) study a European healthcare system where nurses and medical personnel visit patients at home to perform daily tasks. Due to the high number of scheduled visits, completing all tasks within a single day is often impossible. To address this, the authors introduce decision variables that allow tasks to remain uncovered. In a similar setting, Kovacs et al. (2012) examine a maintenance service provider where scheduling all tasks within a day is challenging. Unlike the healthcare case, unassigned tasks are not left unfulfilled but instead outsourced to third-party contractors at a fixed cost. The authors develop an adaptive large neighborhood search heuristic that incorporates destroy-and-repair operators while adjusting task insertion criteria to account for outsourcing costs. If integrating a task into the schedule is infeasible, the model automatically outsources it. Their approach produces high-quality solutions for instances with up to 100 tasks in under two minutes. Zamorano et al. (2018) explore outsourcing in a German ground-handling agency, where check-in workforce scheduling at European airports allows tasks to be outsourced to qualified supervisors or back-office managers at a cost based on task duration. The authors incorporate outsourcing decisions into their model and propose an exact branch-and-price algorithm that efficiently solves instances of up to 60 tasks in short computational times.

1.3.4 Planning horizon

Traditionally, WSRPs have been used for daily operational planning, where scheduling decisions are made for a single day. However, recent research has shifted toward multi-period planning, extending the scheduling horizon to several days, weeks, or even months. This shift introduces a key challenge: tasks must first be assigned to specific days before scheduling workers and constructing routes, significantly increasing problem complexity.

One of the earliest studies incorporating a multi-period planning horizon was conducted by Tricoire et al. (2013), who examined a water distribution company deploying field workers for commercial operations. In this setting, each customer had to be visited exactly once over a set of consecutive days. The authors developed a branch-and-price algorithm capable of solving instances with up to 100 tasks. However, the increased complexity of the problem led to computational times exceeding 24 hours in some cases, preventing optimal solutions for all instances.

In the rehabilitative services sector, Shao et al. (2012) studied an agency that designs weekly schedules for therapists to better align patient demand with therapist skills while minimizing treatment, travel, administrative, and mileage reimbursement costs. They proposed a mixed-integer programming model, which was effective for small instances but computationally prohibitive for larger ones. To address this, they developed a GRASP heuristic that could handle instances with up to 200 tasks. Later, Bard et al. (2014) enhanced this approach by introducing new search neighborhoods, improving solution quality. Their work highlights the importance of multi-period planning in accommodating real-world constraints, such as ensuring patients are first seen by a licensed therapist and managing overtime costs, which typically apply only after 40 hours of regular work.

Recently, multi-period WSRPs have gained significant attention in home health care planning. Liu et al. (2018) studied a nursing agency responsible for planning visits and travel schedules over a weekly horizon. Unlike previous studies, they

incorporated a bi-objective approach, balancing total costs with patient satisfaction. To address this, they applied an ϵ-constrained method capable of generating non-dominated solutions for small instances. Similarly, Mosquera et al. (2019) introduced an optimization-based decision support model that accounts for the trade-off between workforce workload and the number of completed tasks over the planning horizon.

These studies demonstrate that extending WSRPs to multi-period settings enhances operational efficiency by incorporating broader performance metrics. With relatively minimal effort, organizations can achieve significant scheduling improvements, making multi-period planning a valuable direction for future research.

1.3.5 Time windows

In many WSRP applications, tasks must be performed within predefined time windows, representing the period between the earliest and latest allowable service times. These constraints can be hard (strict) or soft, depending on the flexibility permitted in scheduling. Time windows have been widely studied since the seminal work of Solomon (1987), where the authors evaluated heuristic performance across various problem settings. The benchmark instances introduced in that study have since been widely adopted in subsequent research (Lim et al. 2004; Pillac et al. 2013, 2012; Yuan et al. 2015; Zamorano and Stolletz 2017).

Building on this foundation, Akjiratikarl et al. (2007) examined community care providers in the UK, where time windows are determined during the patient assessment process. Depending on task priority, time constraints can be either hard or soft. The authors propose a particle swarm optimization heuristic, claiming improved solution quality. However, their study provides limited details on the benchmark instances used and does not report computational times, making it difficult to fully assess the algorithm's effectiveness.

In practice, respecting time windows depends on unpredictable factors, such as

variable task durations or travel delays due to traffic, which are often overlooked in traditional deterministic models. Recently, researchers have begun incorporating stochastic elements into WSRPs to better reflect real-world uncertainties. Yuan et al. (2015) investigate medical and paramedical service providers, where task durations follow stochastic distributions. They propose a stochastic programming model with recourse, accounting for penalties incurred by late arrivals. Similarly, Zhan and Wan (2018) introduce a tabu search-based approach, modeling uncertain service times through multiple scenarios. Both studies emphasize that ignoring task variability can lead to unrealistic, overly tight schedules, increasing the risk of overtime and reduced service quality.

Future research should focus on developing both exact and heuristic methodologies that effectively integrate stochasticity into WSRPs. Addressing uncertainty in service times and travel conditions will improve solution robustness and ensure more practical implementations in real-world workforce scheduling.

1.3.6 Multi-modal routes

Most WSRP studies focus on single-mode transportation, typically assuming that workers travel exclusively by vehicle. However, in dense urban areas, mobility is often constrained by traffic congestion, limited parking, and accessibility restrictions. To address these challenges, companies increasingly seek solutions that incorporate multi-modal routes, where workers use different transportation modes in a park-and-loop structure.

A multi-modal route consists of a main tour, usually completed by car or van, and subtours performed using alternative modes such as walking, biking, or electric scooters after parking at a designated location. These subtours are often subject to duration constraints, such as battery life for electric scooters. Despite its potential benefits, the integration of multi-modal routing into WSRPs remains underexplored (Cabrera et al. 2025, 2022, 2023; Coindreau et al. 2019; Le Colleter et al. 2023).

In the service sector, Coindreau et al. (2019) analyze a European energy provider where technicians travel between distant customer locations by car, then perform walking subtours to service multiple customers after parking. This structure reduces overall routing costs. Similarly, Cabrera et al. (2022) study an application at ENEDIS, a subsidiary of the French electricity provider EDF. Unlike the previous study, their model incorporates duration constraints for walking subtours and allows routes to start and end at customer locations.

The introduction of multi-modal routes significantly increases problem complexity. Key decisions include where to park, how long subtours should last, and how to balance speed, cost, and accessibility across different transportation modes. However, integrating these features into WSRPs could yield significant benefits, including reduced congestion, lower environmental impact, and more sustainable workforce mobility.

1.3.7 Precedence constraints

Another key yet underexplored feature in WSRPs is the temporal dependence between tasks, where certain tasks cannot begin until others are completed. Incorporating precedence constraints significantly increases problem complexity, requiring careful task sequencing and synchronization.

Goel and Meisel (2013) study a German electricity network maintenance problem where some tasks depend on prior actions. For example, maintenance can only begin after disconnecting the affected part of the network. Efficiently scheduling and synchronizing workers is crucial to minimize downtime. Due to the problem's complexity, the authors employ a decomposition approach, combining exact and heuristic methods. Their approach yields high-quality solutions for instances with up to 100 tasks.

More recently, Pereira et al. (2020) addressed a multi-period WSRP with dependent tasks, where a company must execute interdependent jobs at multiple customer

locations. The presence of precedence constraints and synchronization requirements complicated the design of an efficient local search algorithm. They also tried a mixed-integer programming model that could only solve small instances, limiting its applicability. To overcome these challenges, the authors propose an ant colony optimization metaheuristic, which is independent of local search and requires fewer computational resources. Their method successfully handles instances with up to 60 tasks.

Notwithstanding these advances, precedence constraints remain understudied in WSRPs despite their relevance in maintenance, healthcare, and field services. Existing methods face scalability challenges, and heuristic approaches often overlook problem-specific structures. Future research should focus on developing scalable exact and heuristic methods that better handle task synchronization and temporal dependencies

1.3.8 Constrained resources

Workforce scheduling often assumes unlimited transportation and equipment availability, but in many real-world applications, these resources are constrained. Limited vehicle range, tool availability, and spare parts logistics add significant complexity to WSRPs, requiring specialized optimization approaches.

Villegas et al. (2018) study the case of ENEDIS, a company where workers use both electric and internal combustion vehicles. Electric vehicles are preferred due to their lower operating costs and their contribution to reducing the company's carbon footprint. However, their limited availability and short driving range require careful assignment to the most suitable operations and drivers. The authors emphasize the importance of incorporating recharging stops, noting that failing to do so diminishes the cost savings and operational benefits of using electric vehicles. To address this issue, they develop a two-phase parallel matheuristic capable of solving instances with up to 200 customers.

In maintenance and repair services, workers often require specific tools and spare parts to complete tasks. Pillac et al. (2013) examine a case where technicians start their routes from home with a predefined inventory and must replenish materials at depots before continuing their routes. Similarly, Mathlouthi et al. (2018) focus on electronic transaction equipment maintenance, developing a mixed-integer linear programming model to analyze problem complexity. Initially, their methodology could only solve small instances (fewer than 20 tasks). To improve scalability, they later introduced a matheuristic approach (Mathlouthi et al. 2021a) and a branch-and-price algorithm (Mathlouthi et al. 2021b), successfully handling instances with up to 200 tasks.

Resource limitations are a crucial yet often overlooked aspect of WSRPs. Future research should focus on scalable methodologies that efficiently integrate transportation constraints, equipment availability, and replenishment strategies, ensuring practical solutions for real-world workforce scheduling.

1.3.9 Synthesis and key takeaways

Table 1.1 summarizes the key features explored in WSRP literature. Column 1 lists the references, while Column 2 provides the acronym used to describe each problem variant. Columns 3 to 11 indicate the inclusion of specific features in each study, marked with an "x" when present and left empty otherwise. This table serves two important purposes. First, it allows researchers working on WSRPs to quickly identify relevant studies that share similar constraints or problem settings, providing a valuable baseline for future work. Second, it highlights well-explored areas with limited opportunities for novel contributions, as well as underrepresented topics where further research is needed.

The majority of WSRP studies include time windows (48 out of 58) and skill compatibility (44 out of 58), indicating their central role in problem modeling. In contrast, multi-modal routes (5 studies) and precedence constraints (3 studies) remain

largely unexplored. Additionally, most existing works that incorporate multi-modal routes assume a homogeneous workforce and are limited to single day planning. This gap highlights an opportunity to develop algorithms that integrate both multi-modal routing and a multi-period planning horiozon in WSRPs.

Another key observation is the high variability in acronyms used to refer to WSRP variants. Even when two studies consider the same features, their chosen acronyms often differ, creating challenges for both researchers and practitioners. First, this inconsistency makes it difficult to locate relevant studies without prior knowledge of how features are incorporated into the problem title. Second, it increases the risk of redundant research, as similar works may go unnoticed due to naming discrepancies.

To address this issue, we propose the following standardization guidelines. First, since time windows and skill compatibility are commonly included in WSRPs, they should not be explicitly mentioned in problem titles or acronyms. Second, researchers should avoid embedding specific job roles in problem acronyms, opting instead for a more general designation that applies across different workforce types (e.g., replacing "technician scheduling and routing problem" with "workforce scheduling and routing problem"). Finally, additional problem features should be appended to the acronym using hyphen-separated abbreviations. For example, the MFSRP acronym used in Tricoire et al. (2013) (which includes team building, multiple periods, and time windows) would be standardized as WSRP-MPTB under these guidelines.

Table 1.2 categorizes the solution methodologies used in WSRP research. Column 1 lists the methodologies, Column 2 cites the studies that applied each approach, and Column 3 indicates the total number of articles using each method. A final category includes methodologies that have been used only once. Notably, eight studies employ a branch-and-price approach, making it the most widely used exact method. In contrast, only four studies present a mixed-integer programming model as their main contribution. Overall, just 17 out of 58 studies propose exact

approaches, and many of these are limited to solving small instances. This highlights the need for further research on exact algorithms that can efficiently handle larger-scale WSRP instances.

Table 1.1: Features of WSRPs found in the literature

Reference	Acronym	T.W.	S.C.	T.B.	O.	M.P.	C.R.	M.M.	P.C
Tsang and Voudouris (1997)	WSP	x	x		x				
Lim et al. (2004)	MAPTW	x		x					
Li et al. (2005)	MAPTWTC	x	x	x					
Akjiratikarl et al. (2007)	HCWS	x							
Bredström and Rönnqvist (2008)	VRSPTW	x		x					
Dohn and Sevel (2008)	HCCSP	x	x	x	x				
Dohn et al. (2009)	MAPTWTC	x	x	x					
Mendoza et al. (2009)	DVRP					x	x		
Tricoire et al. (2013)	MFSRP	x		x		x			
Misir et al. (2011)	SPRR	x	x			x			
Lau and Gunawan (2012)	PSP	x							
Rasmussen et al. (2012)	HCCSP	x	x	x	x				x
Kovacs et al. (2012)	TRSP	x	x	x	x				
Pillac et al. (2012)	D-TRSP	x	x				x		
Shao et al. (2012)	TRSP	x	x			x			
Günther and Nissen (2013)	WSRP	x	x		x				
Pillac et al. (2013)	TRSP	x	x				x		
Goel and Meisel (2013)	WSRP			x					x
Bard et al. (2014)	TRSP	x	x			x			
Yuan et al. (2015)	HHCSR	x	x						
Chen et al. (2016)	TRSP-EST		x			x			
Braekers et al. (2016)	HCSRP	x	x						

Legend: T.W. - Time windows; S.C. - Skill compatibility; T.B. - Team building; O. - Outsourcing; M.P. - Multi-period; C.R. - Constrained resources; M.M. - Multi-modal; P.C. - Precendence constraints.

Table 1.1: (Continued)

Reference	Acronym	T.W.	S.C.	T.B.	O.	M.P.	C.R.	M.M.	P.C
Xie et al. (2017)	WSRP	x	x		x				
Zamorano and Stolletz (2017)	MPTRSP	x	x	x		x			
Villegas et al. (2018)	TRSP-CEV	x	x				x		
Liu et al. (2018)	HCCSP	x	x	x		x			
Mathlouthi et al. (2018)	TRSP	x	x		x		x		
Zamorano et al. (2018)	-	x	x	x	x				
Zhan and Wan (2018)	RASTA	x		x					
Algethami et al. (2019)	WSRP	x	x		x				
Anoshkina and Meisel (2019)	CTRP		x	x					
Mosquera et al. (2019)	HCCSP	x	x			x			
Coindreau et al. (2019)	VRPTR	x					x	x	
Grenouilleau et al. (2019)	HHCRSP	x	x				x	x	
Grenouilleau et al. (2020)	HCS-PV	x	x				x	x	
Pekel (2020)	TRSP	x	x	x		x			
Çakırgil et al. (2020)	WSRP	x	x	x	x		x		
Pereira et al. (2020)	MWSRPDT			x		x			x
Zhou et al. (2020)	WSRP	x	x		x				
Guastaroba et al. (2021)	MPWSRP	x	x		x	x			
Mathlouthi et al. (2021b)	TRSP	x	x		x		x		
Mathlouthi et al. (2021a)	TRSP	x	x		x		x		
Cabrera et al. (2022)	DOPLRP						x	x	
Gu et al. (2022)	WSRP	x	x		x				

Legend: T.W. - Time windows; S.C. - Skill compatibility; T.B. - Team building; O. - Outsourcing; M.P. - Multi-period; C.R. - Constrained resources; M.M. - Multi-modal; P.C. - Precendence constraints.

Table 1.1: (Continued)

Reference	Acronym	T.W.	S.C.	T.B.	O.	M.P.	C.R.	M.M.	P.C
Cabrera et al. (2023)	PLRP						x	x	
Le Colleter et al. (2023)	PLRP-PS						x	x	
Naderi et al. (2023)	HHCPS	x	x			x			
Clapper et al. (2023)	HHCRSP	x	x						
Bazirha et al. (2023)	HHCRSP	x	x	x					
Qiu et al. (2023)	MSTRP	x	x	x		x	x		
Su et al. (2023)	MAVRP-SQTW	x	x	x			x		
Nowak and Szufel (2024)	TRSP		x			x	x		
Huang et al. (2024)	MRVRPTW-MSM	x	x				x		
Rastegar et al. (2024)	TRSP	x					x		
Delavernhe et al. (2024)	OSTRP	x	x			x	x		
Du and Wang (2024)	HHCRSP	x	x						
Li and Liu (2025)	TRSP		x	x					
Cabrera et al. (2025)	WSRP-PL	x	x	x	x		x	x	
Total		48	44	22	16	18	21	5	3

Legend: T.W. - Time windows; S.C. - Skill compatibility; T.B. - Team building; O. - Outsourcing; M.P. - Multi-period; C.R. - Constrained resources; M.M. - Multi-modal; P.C. - Precendence constraints.

Table 1.2: Solution algorithms and methodologies.

Methodology	References	# Articles
Simulated annealing	Lim et al. (2004), Li et al. (2005), Bazirha et al. (2023).	3
Particle swarm optimization	Akjiratikarl et al. (2007), Günther and Nissen (2013), Pekel (2020).	3
Fast local search	Solomon (1987), Tsang and Voudouris (1997).	2
Adaptive large neighborhood search	Kovacs et al. (2012), Pillac et al. (2012), Pillac et al. (2013), Goel and Meisel (2013), Coindreau et al. (2019).	5
Branch-and-price	Dohn and Sevel (2008), Dohn et al. (2009), Tricoire et al. (2013), Rasmussen et al. (2012), Yuan et al. (2015), Zamorano and Stolletz (2017), Zamorano et al. (2018), Mathlouthi et al. (2021b).	8
Branch-price-and-cut	Cabrera et al. (2023), Su et al. (2023), Du and Wang (2024), Huang et al. (2024), Cabrera et al. (2025).	5
Mixed integer program	Lau and Gunawan (2012), Shao et al. (2012), Bard et al. (2014), Mathlouthi et al. (2018).	4
Matheuristic	Villegas et al. (2018), Çakırgil et al. (2020), Mathlouthi et al. (2021a), Cabrera et al. (2022), Guastaroba et al. (2021), Grenouilleau et al. (2020).	6
Other (heuristic)	Bredström and Rönnqvist (2008), Mendoza et al. (2009), Misir et al. (2011), Chen et al. (2016), Braekers et al. (2016), Xie et al. (2017), Liu et al. (2018), Zhan and Wan (2018), Grenouilleau et al. (2019), Algethami et al. (2019), Anoshkina and Meisel (2019), Mosquera et al. (2019), Pereira et al. (2020), Zhou et al. (2020), Gu et al. (2022), Le Colleter et al. (2023), Naderi et al. (2023), Clapper et al. (2023), Qiu et al. (2023), Nowak and Szufel (2024), Delavernhe et al. (2024), Rastegar et al. (2024), Li and Liu (2025).	23

1.4 From theory to practice

The previous section reviewed various exact and heuristic methodologies developed to solve different WSRP variants. However, little discussion has been devoted to the practical implementation of these solutions in real-world settings. A key example of this challenge is presented by Holland et al. (2017), who examines the deployment of an optimization system for UPS to organize its pick-up and delivery operations.

Although the study focuses on parcel delivery, it illustrates how neglecting practical considerations can lead to implementation failure. Initially, UPS adopted a

metaheuristic-based approach that achieved significant savings in laboratory tests. However, when applied in practice, the generated routes proved difficult for drivers to follow, leading to low adoption rates. After investigating the issue, UPS revised its methodology to generate consistent, easy-to-follow routes that also promoted safer driving behavior, avoiding unnecessary zigzags and left turns on busy streets. The revised approach was successfully implemented, generating annual savings of up to 300 million dollars highlighting the critical role of practical considerations in WSRP applications.

Beyond operational feasibility, several practical factors influence the effectiveness of WSRP solutions. The following subsections examine research focused on integrating continuity, visual attractiveness, and worker behavior into scheduling and routing problems. Each subsection begins with a brief explanation of the practical consideration, followed by a review of related works and guidelines for future research.

1.4.1 Continuity

Continuity in WSRPs refers to the consistent assignment of the same worker to a given customer over multiple visits. This feature is particularly relevant in health care, where continuity between patients and medical professionals has been linked to improved patient outcomes.

Russell et al. (2011) analyze the relationship between service continuity and patient outcomes, proposing a continuity measure based on the total number of patient visits and the number of visits conducted by the same nurse. They evaluate its impact on three key outcomes: hospitalization rates, emergency room visits, and quality of daily living. Using a logistic regression model on a dataset of approximately 60,000 patients in the UK, their findings indicate that greater continuity is associated with better patient outcomes. However, their analysis does not account for correlations within patient groups (e.g., based on location or financial status),

which may lead to underestimated variance and potential bias in their results. Similarly, Facchinetti et al. (2020) conduct a meta-analysis on patient readmission, concluding that continuity of care prevents service quality deterioration and reduces readmissions, a costly and undesirable event.

From a managerial perspective, Gjevjon et al. (2013) explore service continuity in health care organizations through semi-structured interviews with managers from small, medium, and large providers in Norway. Their findings reveal common challenges, including budget constraints, staff shortages, and heavy workloads, which often limit the feasibility of ensuring continuity. While managers recognize its importance and consider patient preferences when scheduling workers, operational constraints frequently prevent full implementation. This highlights the trade-off between continuity and logistical feasibility, underscoring the need for workforce scheduling models that balance efficiency with patient-centered care.

As discussed, continuity plays a crucial role in improving patient outcomes by fostering trust and enhancing service quality. Despite its importance, this feature has been largely overlooked in WSRP methodologies. Future research should focus on incorporating constraints that ensure continuity in scheduling. For instance, in multi-period planning models (see Section 1.3.4), constraints could enforce that a given customer is always assigned to the same subset of workers. Additionally, heuristic approaches could introduce operators that prioritize continuity, making it a key factor in workforce scheduling optimization.

1.4.2 Visual attractiveness

The debate over what constitutes a good-looking versus bad-looking route has gained significant attention in routing research. Rossit et al. (2019) present a comprehensive literature review on this topic, concluding that visually attractive routes should be simple, compact, and non-overlapping. Simplicity relates to intuitiveness, compactness ensures that customers within a route are geographically close, and

non-overlapping routes prevent workers from covering the same area multiple times. The authors also evaluate metrics for assessing these features, such as the number of route crossings, which could be integrated into WSRP methodologies.

Similarly, Hollis and Green (2012) analyze metrics for visual attractiveness and operational robustness using data from a beverage distribution company. Their findings indicate that prioritizing visual attractiveness over cost minimization leads to better service quality, as it allows greater flexibility in revisiting customers when needed. They argue that visually structured routes improve operational efficiency by making adjustments easier, particularly in unpredictable environments. Other studies, such as Sahoo et al. (2005) and Shao et al. (2014), explore visual attractiveness from the perspective of routing complexity.

In the WSRP context, considering visual attractiveness could lead to significant operational improvements. As highlighted by Rossit et al. (2019), visually structured routes enhance solution acceptance, making them more practical for real-world implementation rather than just academic models. Additionally, compact routes improve adaptability in dynamic environments, where traffic congestion or road closures require real-time adjustments. For example, in multi-modal WSRPs, where workers park before completing subtours, a compact routing structure ensures alternative paths are available if a designated parking spot is occupied. Researchers could integrate this feature by imposing constraints that limit the maximum distance between consecutive customers, ensuring better flexibility and responsiveness.

1.4.3 Workers behavior

Regardless of the algorithm or methodology used to generate WSRP solutions, their effectiveness ultimately depends on worker behavior. However, as noted by Srivatsa Srinivas and Gajanand (2017), researchers and practitioners often overlook behavioral factors when designing scheduling and routing plans. Worker behavior is influenced by various factors, including traffic congestion and task-related fatigue.

Ma et al. (2014) analyze the impact of traffic alerts on driver decisions, specifically focusing on variable message signs (VMS) in Beijing. They conducted a set of interviews and used a multinomial logit model, where the dependent variable is the driver's response and the independent variables include demographic factors and VMS-related characteristics, such as message intensity and display mode. Their findings indicate a strong relationship between socioeconomic characteristics and route choice behavior. Specifically, workers driving private vehicles are more likely to take alternative routes in response to traffic jams, while those driving company-owned vehicles tend to adhere to the original route. While the study provides well-defined categories and a clear sampling methodology, its applicability outside Beijing remains uncertain due to differences in traffic management systems.

Similarly, Zhou et al. (2014) examine VMS impact through prospect theory, demonstrating how different worker personalities and characteristics influence route choices. Other studies exploring the relationship between worker conditions and decision-making include Haughton (2009), Gastaldi et al. (2014), and Morris and Hirsch (2016)

1.5 Concluding remarks

This work presents a comprehensive literature review of the workforce scheduling and routing problem (WSRP) and its many variants, which have garnered increasing attention from researchers over the past decades. Our review categorizes the most relevant studies according to eight key features found in the literature. For each feature, we discuss real-world applications and examine how its inclusion affects the methodologies developed for solving WSRPs. By outlining the past, present, and future of this problem, we aim to inspire further research and the development of new methodologies in this field.

One major challenge identified in this review is the lack of a recognized state-of-the-art algorithm for solving the WSRP. This gap stems from the limited direct

comparisons between existing methodologies. To advance the field, a comprehensive benchmark study evaluating multiple approaches on a common set of instances is essential. We also urge researchers to avoid introducing new algorithms without assessing their performance against previous methods, both in terms of solution quality and computational efficiency.

Beyond methodological advancements, this review highlights practical considerations that should be incorporated into WSRP models to enhance real-world applicability. We conclude that future research would benefit from integrating factors such as continuity, visual attractiveness, and worker behavior into solution methods.

References

Akjiratikarl, C., P. Yenradee, and P. R. Drake (2007). "PSO-based algorithm for home care worker scheduling in the UK". *Computers & Industrial Engineering* 53.4, pp. 559–583.

Algethami, H., A. Martínez-Gavara, and D. Landa-Silva (2019). "Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem". *Journal of Heuristics* 25, pp. 753–792.

Anoshkina, Y. and F. Meisel (2019). "Technician teaming and routing with service-, cost-and fairness-objectives". *Computers & Industrial Engineering* 135, pp. 868–880.

Bard, J. F., Y. Shao, and A. I. Jarrah (2014). "A sequential GRASP for the therapist routing and scheduling problem". *Journal of Scheduling* 17, pp. 109–133.

Bazirha, M., R. Benmansour, and A. Kadrani (2023). "An efficient two-phase heuristic for the home care routing and scheduling problem". *Computers & Industrial Engineering* 181, p. 109329.

Braekers, K., R. F. Hartl, S. N. Parragh, and F. Tricoire (2016). "A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience". *European Journal of Operational Research* 248.2, pp. 428–443.

Bredström, D. and M. Rönnqvist (2008). "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints". *European Journal of Operational Research* 191.1, pp. 19–31.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2025). "The workforce scheduling and routing problem with park-and-loop". *Networks* 85.1, pp. 38–60.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2022). "The doubly open park-and-loop routing problem". *Computers & Operations Research* 143, p. 105761.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2023). "Solving the park-and-loop routing problem by branch-price-and-cut". *Transportation Research Part C: Emerging Technologies* 157, p. 104369.

Çakırgil, S., E. Yücel, and G. Kuyzu (2020). "An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems". *Computers & Operations Research* 118, p. 104908.

Chen, X., B. W. Thomas, and M. Hewitt (2016). "The technician routing problem with experience-based service times". *Omega* 61, pp. 49–61.

Clapper, Y., J. Berkhout, R. Bekker, and D. Moeke (2023). "A model-based evolutionary algorithm for home health care scheduling". *Computers & Operations Research* 150, p. 106081.

Coindreau, M.-A., O. Gallay, and N. Zufferey (2019). "Vehicle routing with transportable resources: Using carpooling and walking for on-site services". *European Journal of Operational Research* 279, pp. 996–1010.

Cordeau, J.-F., G. Laporte, F. Pasin, and S. Ropke (2010). "Scheduling technicians and tasks in a telecommunications company". *Journal of Scheduling* 13, pp. 393–409.

Costa, L., C. Contardo, and G. Desaulniers (2019a). "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing". *Transportation Science* 53, pp. 946–985.

Delavernhe, F., B. Castanier, C. Guéret, and J. E. Mendoza (2024). "The joint maintenance operation selection and technician routing problem". *Computers & Operations Research* 167, p. 106667.

Desrosiers, J., M. Lübbecke, G. Desaulniers, and J. B. Gauthier (2024). *Branch-and-Price*. Les Cahiers du GERAD G-2024-36. GERAD, Montréal QC H3T 2A7, Canada: Groupe d'études et de recherche en analyse des décisions, pp. 1–657.

Dohn, A., E. Kolind, and J. Clausen (2009). "The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach". *Computers & Operations Research* 36.4, pp. 1145–1157.

Dohn, A. and M. Sevel (2008). "The home care crew scheduling problem". *Conference Proceedings of the 1st International Conference on Applied Operational Research*.

Du, J. and X. Wang (2024). "A branch-and-price-and-cut algorithm for the home health care routing and scheduling problem with multiple prioritized time windows". *Computers & Operations Research* 170, p. 106749.

Estellon, B., F. Gardi, and K. Nouioua (2009). "High-performance local search for task scheduling with human resource allocation." *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics* 5752, pp. 1–16.

Facchinetti, G., D. D'Angelo, M. Piredda, T. Petitti, M. Matarese, A. Oliveti, and M. G. De Marinis (2020). "Continuity of care interventions for preventing hospital readmission of older people with chronic diseases: A meta-analysis". *International Journal of Nursing Studies* 101, p. 103396.

Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen (2004). "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". *Networks* 44.3, pp. 216–229.

Fırat, M. and C. A. Hurkens (2012). "An improved MIP-based approach for a multi-skill workforce scheduling problem". *Journal of Scheduling* 15.3, pp. 363–380.

Gastaldi, M., R. Rossi, and G. Gecchele (2014). "Effects of driver task-related fatigue on driving performance". *Procedia-Social and Behavioral Sciences* 111, pp. 955–964.

Gjevjon, E. R., T. I. Romøren, B. Ø. Kjøs, and R. Hellesø (2013). "Continuity of care in home health-care practice: two management paradoxes". *Journal of Nursing Management* 21.1, pp. 182–190.

Goel, A. and F. Meisel (2013). "Workforce routing and scheduling for electricity network maintenance with downtime minimization". *European Journal of Operational Research* 231.1, pp. 210–228.

Grenouilleau, F., N. Lahrichi, and L.-M. Rousseau (2020). "New decomposition methods for home care scheduling with predefined visits". *Computers & Operations Research* 115, p. 104855.

Grenouilleau, F., A. Legrain, N. Lahrichi, and L.-M. Rousseau (2019). "A set partitioning heuristic for the home health care routing and scheduling problem". *European Journal of Operational Research* 275.1, pp. 295–303.

Gu, H., Y. Zhang, and Y. Zinder (2022). "An efficient optimisation procedure for the workforce scheduling and routing problem: Lagrangian relaxation and iterated local search". *Computers & Operations Research* 144, p. 105829.

Guastaroba, G., J.-F. Côté, and L. C. Coelho (2021). "The multi-period workforce scheduling and routing problem". *Omega* 102, p. 102302.

Günther, M. and V. Nissen (2013). "Application of Particle Swarm Optimization to the British Telecom Workforce Scheduling Problem". *Proceedings of the 9th international conference on the practice and theory of automated timetabling (PATAT 2012)*, pp. 242–256.

Hashimoto, H., S. Boussier, M. Vasquez, and C. Wilbaut (2011). "A GRASP-based approach for technicians and interventions scheduling for telecommunications". *Annals of Operations Research* 183, pp. 143–161.

Haughton, M. A. (2009). "An alternative tactic to deal with the contingency of driver absenteeism". *Journal of the Operational Research Society* 60.9, pp. 1207–1220.

Holland, C., J. Levis, R. Nuggehalli, B. Santilli, and J. Winters (2017). "UPS optimizes delivery routes". *Interfaces* 47.1, pp. 8–23.

Hollis, B. and P. Green (2012). "Real-life vehicle routing with time windows for visual attractiveness and operational robustness". *Asia-Pacific Journal of Operational Research* 29.04, p. 1250017.

Kovacs, A. A., S. N. Parragh, K. F. Doerner, and R. F. Hartl (2012). "Adaptive large neighborhood search for service technician routing and scheduling problems". *Journal of Scheduling* 15, pp. 579–600.

Lau, H. C. and A. Gunawan (2012). "The Patrol Scheduling Problem". *Practice and Theory of Automated Timetabling*.

Le Colleter, T., D. Dumez, F. Lehuédé, and O. Péton (2023). "Small and large neighborhood search for the park-and-loop routing problem with parking selection". *European Journal of Operational Research* 308.3, pp. 1233–1248.

Li, Y., A. Lim, and B. Rodrigues (2005). "Manpower allocation with time windows and job-teaming constraints". *Naval Research Logistics (NRL)* 52.4, pp. 302–311.

Lim, A., B. Rodrigues, and L. Song (2004). "Manpower allocation with time windows". *Journal of the Operational Research Society* 55.11, pp. 1178–1186.

Liu, M., D. Yang, Q. Su, and L. Xu (Sept. 2018). "Bi-objective approaches for home healthcare medical team planning and scheduling problem". *Computational and Applied Mathematics* 37 (4), pp. 4443–4474.

Ma, Z., C. Shao, Y. Song, and J. Chen (2014). "Driver response to information provided by variable message signs in Beijing". *Transportation research part F: traffic psychology and behaviour* 26, pp. 199–209.

Mathlouthi, I., M. Gendreau, and J.-Y. Potvin (2018). "Mixed integer linear programming for a multi-attribute technician routing and scheduling problem". *INFOR: Information Systems and Operational Research* 56.1, pp. 33–49.

Mathlouthi, I., M. Gendreau, and J.-Y. Potvin (2021a). "A metaheuristic based on Tabu search for solving a technician routing and scheduling problem". *Computers & Operations Research* 125, p. 105079.

Mathlouthi, I., M. Gendreau, and J.-Y. Potvin (2021b). "Branch-and-price for a multi-attribute technician routing and scheduling problem". *Operations Research Forum* 2, pp. 1–35.

Mendoza, J. E., A. L. Medaglia, and N. Velasco (2009). "An evolutionary-based decision support system for vehicle routing: The case of a public utility". *Decision Support Systems* 46.3, pp. 730–742.

Misir, M., P. Smet, K. Verbeeck, and G. Vanden Berghe (2011). "Security personnel routing and rostering: a hyper-heuristic approach". *Proceedings of the 3rd International Conference on Applied Operational Research*. Vol. 3. Tadbir; Canada, pp. 193–205.

Morris, E. A. and J. A. Hirsch (2016). "Does rush hour see a rush of emotions? Driver mood in conditions likely to exhibit congestion". *Travel Behaviour and Society* 5, pp. 5–13.

Mosquera, F., P. Smet, and G. V. Berghe (2019). "Flexible home care scheduling". *Omega* 83, pp. 80–95.

Naderi, B., M. A. Begen, G. S. Zaric, and V. Roshanaei (2023). "A novel and efficient exact technique for integrated staffing, assignment, routing, and scheduling of home care services under uncertainty". *Omega* 116, p. 102805.

Nowak, M. and P. Szufel (2024). "Technician routing and scheduling for the sharing economy". *European Journal of Operational Research* 314.1, pp. 15–31.

Paraskevopoulos, D. C., G. Laporte, P. P. Repoussis, and C. D. Tarantilis (2017). "Resource constrained routing and scheduling: Review and research prospects". *European Journal of Operational Research* 263.3, pp. 737–754.

Pekel, E. (2020). "Solving technician routing and scheduling problem using improved particle swarm optimization". *Soft Computing* 24.24, pp. 19007–19015.

Pereira, D. L., J. C. Alves, and M. C. de Oliveira Moreira (2020). "A multiperiod workforce scheduling and routing problem with dependent tasks". *Computers & Operations Research* 118, p. 104930.

Pillac, V., C. Gueret, and A. L. Medaglia (2013). "A parallel matheuristic for the technician routing and scheduling problem". *Optimization Letters* 7.7, pp. 1525–1535.

Pillac, V., C. Guéret, and A. L. Medaglia (2012). "On the dynamic technician routing and scheduling problem". PhD thesis. Ecole des Mines de Nantes.

Qiu, H., J. Wang, D. Wang, and Y. Yin (2023). "Service-oriented multi-skilled technician routing and scheduling problem for medical equipment maintenance with sudden breakdown". *Advanced Engineering Informatics* 57, p. 102090.

Rasmussen, M. S., T. Justesen, A. Dohn, and J. Larsen (2012). "The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies". *European Journal of Operational Research* 219.3, pp. 598–610.

Rastegar, M., H. Karimi, and H. Vahdani (2024). "Technicians scheduling and routing problem for elevators preventive maintenance". *Expert Systems with Applications* 235, p. 121133.

Rossit, D. G., D. Vigo, F. Tohmé, and M. Frutos (2019). "Visual attractiveness in routing problems: A review". *Computers & Operations Research* 103, pp. 13–34.

Russell, D., R. J. Rosati, P. Rosenfeld, and J. M. Marren (2011). "Continuity in home health care: is consistency in nursing personnel associated with better patient outcomes?" *Journal for Healthcare Quality* 33.6, pp. 33–39.

Sahoo, S., S. Kim, B.-I. Kim, B. Kraas, and A. Popov Jr (2005). "Routing optimization for waste management". *Interfaces* 35.1, pp. 24–36.

Shao, J., L. Kulik, E. Tanin, and L. Guo (2014). "Travel distance versus navigation complexity: A study on different spatial queries on road networks". *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1791–1794.

Shao, Y., J. F. Bard, and A. I. Jarrah (2012). "The therapist routing and scheduling problem". *IIE Transactions* 44.10, pp. 868–893.

Solomon, M. M. (1987). "Algorithms for the vehicle routing and scheduling problems with time window constraints". *Operations Research* 35.2, pp. 254–265.

Srivatsa Srinivas, S. and M. Gajanand (2017). "Vehicle routing problem and driver behaviour: a review and framework for analysis". *Transport Reviews* 37.5, pp. 590–611.

Su, X., G. Xu, N. Huang, and H. Qin (2023). "A branch-and-price-and-cut for the manpower allocation and vehicle routing problem with staff qualifications and time windows". *Advanced Engineering Informatics* 57, p. 102093.

Tricoire, F., N. Bostel, P. Dejax, and P. Guez (2013). "Exact and hybrid methods for the multiperiod field service routing problem". *Central European Journal of Operations Research* 21, pp. 359–377.

Tsang, E. and C. Voudouris (1997). "Fast local search and guided local search and their application to British Telecom's workforce scheduling problem". *Operations Research Letters* 20.3, pp. 119–127.

Uchoa, E., A. Pessoa, and L. Moreno (2024). *Optimizing with Column Generation: Advanced Branch-Cut-and-Price Algorithms (Part I)*. Tech. rep. L-2024-3. Cadernos do LOGIS-UFF, Universidade Federal Fluminense, Engenharia de Produção.

Vidal, T., G. Laporte, and P. Matl (2020). "A concise guide to existing and emerging vehicle routing problem variants". *European Journal of Operational Research* 286.2, pp. 401–416.

Villegas, J., C. Prins, A. Medaglia, and N. Velasco (2010). "GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots". *Engineering Applications of Artificial Intelligence* 23, pp. 780–794.

Villegas, J., C. Guéret, J. E. Mendoza, and A. Montoya (June 2018). "The technician routing and scheduling problem with conventional and electric vehicle". working paper or preprint. URL: `https://hal.archives-ouvertes.fr/hal-01813887`.

Xie, F., C. N. Potts, and T. Bektaş (2017). "Iterated local search for workforce scheduling and routing problems". *Journal of Heuristics* 23, pp. 471–500.

Yuan, B., R. Liu, and Z. Jiang (2015). "A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill

requirements". *International Journal of Production Research* 53.24, pp. 7450–7464.

Zamorano, E., A. Becker, and R. Stolletz (2018). "Task assignment with start time-dependent processing times for personnel at check-in counters". *Journal of Scheduling* 21, pp. 93–109.

Zamorano, E. and R. Stolletz (2017). "Branch-and-price approaches for the multi-period technician routing and scheduling problem". *European Journal of Operational Research* 257.1, pp. 55–68.

Zhan, Y. and G. Wan (2018). "Vehicle routing and appointment scheduling with team assignment for home services". *Computers & Operations Research* 100, pp. 1–11.

Zhou, L., S. Zhong, S. Ma, and N. Jia (2014). "Prospect theory based estimation of drivers' risk attitudes in route choice behaviors". *Accident Analysis & Prevention* 73, pp. 1–11.

Zhou, Y., M. Huang, H. Wu, G. Chen, and Z. Wang (2020). "Iterated local search with hybrid neighborhood search for workforce scheduling and routing problem". *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, pp. 478–485.

Chapter 2

Solving the Park-and-loop Routing Problem by Branch-price-and-cut

Abstract

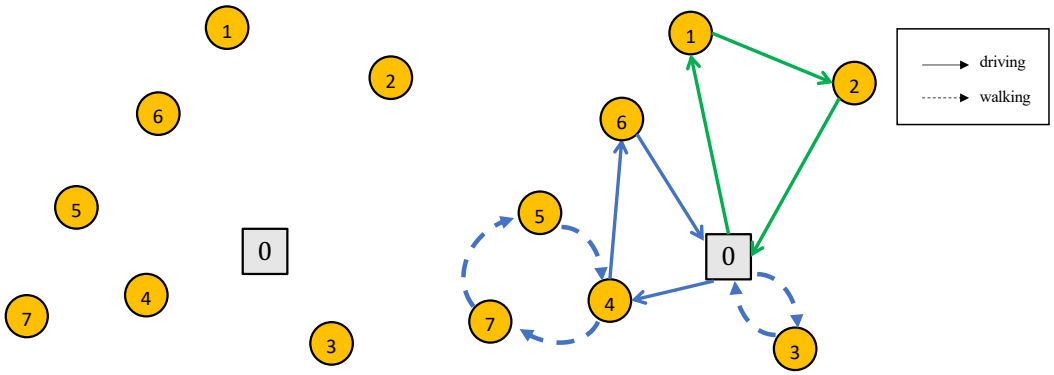
The park-and-loop routing problem is a variation of the vehicle routing problem in which routes include a main tour that is completed using a vehicle and subtours that are carried out on foot after parking the vehicle. Additionally, the route duration and total walking distance are bounded. To solve the problem, we propose an exact solution method based on the branch-price-and-cut framework. In particular, our method uses problem-specific components to solve the pricing problem. We report on computational experiments carried out on a standard set of 40 instances with up to 50 customers. The results show that our method delivers solutions that compare favorably to existing metaheuristic algorithms, matching all previously best-known solutions and improving 11 of them in reasonable computational times. Moreover, our method provides optimality certificates for 39 out of the 40 instances.

2.1 Introduction

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed multigraph where $\mathcal{V}$ is the set of vertices and $\mathcal{A}$ denotes the set of directed arcs. The set of vertices comprises a depot 0 and the set of customers $C = \{1, ..., C\}$. Each customer $i \in C$ has a service time s_i. There are k homogeneous workers available to serve the customers. These workers can either drive or walk between locations. Accordingly, the set of arcs $\mathcal{A} = \{(i, j)^m | m = \{d, w\}\}$ contains two arcs between each pair of vertices, namely, a driving arc (d) and a walking arc (w). Each arc $(i, j)^m \in \mathcal{A}$ has two main attributes: the distance δ_{ij} and the time η_{ij}. Each worker has a maximum daily walking distance ζ and is hired for a full working day lasting ϕ units of time. The maximum distance that a worker may walk between two points is θ. The fixed cost of hiring a worker is c^f. Additionally, there is a variable cost associated with driving (c^v per unit of distance). The park-and-loop routing problem (PLRP) consists in finding a set of least cost routes (starting and ending at the depot) while ensuring that: each customer is served precisely once; the total duration of each route does not exceed the working day duration; and the distance walked by any worker does not exceed the limit. Because it is a generalization of the well-known vehicle routing problem (VRP), the PLRP is NP-hard.

In order to serve the set of customers three types of routes can be designed: pure vehicle routes performed by a worker driving between customers; pure walking routes performed by a worker walking between customers; and finally park-and-loop routes that are vehicle routes with walking subtours. The number of customers in a walking subtour is not constrained. Figure 2.1 shows a feasible solution to a small PLRP instance with 7 customers. In this solution, customers 1 and 2 are served by a worker driving a vehicle (i.e., a pure vehicle route), while customers 3 to 7 are served by a worker following a park-and-loop route. More specifically, the worker leaves (i.e., parks) the vehicle at the depot and walks to serve customer 3. The worker then walks back to the depot to pick up the vehicle and drives to customer 4. After

serving customer 4, the worker parks the vehicle and walks to serve customers 5 and 7. The worker then walks back to customer 4. Finally, the worker drives to customer 6 and then returns to the depot.



(a) Node locations (b) PLRP solution

Figure 2.1: PLRP instance and solution example.

The PLRP is closely related to the two-echelon last-mile delivery problem (2E-LMDP) discussed by Martinez-Sykora et al. (2020); a variant of the traveling salesman problem (TSP) consisting in finding a single park-and-loop route to serve a set of customers. The authors propose an exact branch-and-cut algorithm capable of solving instances with up to 30 customers. As opposed to the PLRP, the number of walking subtours starting at a given parking spot is limited to one. More recently, Reed et al. (2024) introduced the capacitated delivery problem with parking (CDPP). This problem extends the 2E-LMDP by allowing the worker to perform an unlimited number of walking subtours starting at the same parking spot. The authors propose a mixed integer programming (MIP) formulation that can solve instances with up to 50 customers. In addition, they describe a two-step heuristic that can handle instances with up to 100 customers. However, the authors limit the number of customers in a walking subtour to three. As stated previously, in the PLRP, the number of customers in a walking subtour is not constrained.

Another related problem is the two-echelon vehicle routing problem with time

windows and mobile satellites (2EVRP-TM) introduced by Li et al. (2020). In this variant of the VRP each vehicle (i.e., truck) carries a fixed number of drones. These drones are launched either at the depot or at customer locations after parking the vehicle. After all the dispatched drones return to the vehicle, the route continues. To solve this problem, the authors propose an adaptive large neighborhood search (ALNS) heuristic that can handle instances with up to 100 customers. More recently, Zhou et al. (2023) presented a branch-and-price algorithm for solving a problem similar to the 2EVRP-TM in which the number of drones in each vehicle is treated as a decision variable. In this problem variant, once a drone returns to the truck, it may immediately depart to deliver a parcel to another customer. The authors test their algorithm on instances with up to 25 customers. As opposed to the PLRP, in the 2EVRP-TM it is possible that two (or more) subtours are carried out at the same time (by different drones) departing from the same parking spot. Another stream of research has focused on variants of the vehicle routing problem with drones (VRPD). In this setting, drones may be launched at one location and retrieved at another. This drastically changes the structure of the routes, which no longer follow a regular park-and-loop structure, and introduces synchronization issues that are not present in the PLRP. The interested reader is referred to Tamke and Buscher (2021) and Yin et al. (2023) for more details.

The PLRP is also related to the truck and trailer routing problem (TTRP) introduced by Semet (1995). This variant of the VRP considers a fleet of trucks pulling trailers to serve a set of customers. The problem also considers a set of decoupling locations (i.e., parking places), where trailers can be detached as some of the customers are only accessible by the truck without the trailer. Most of the work on the TTRP has focused on heuristic algorithms (Chao 2002; Derigs et al. 2013; Lin et al. 2009; Sheuerer 2006; Villegas et al. 2013, 2011). These methods are capable of providing high quality solutions for instances with up to 150 customers. In the TTRP, routes are constrained by the combined capacity of the truck and the trailer. In contrast, routes in the PLRP are constrained by the maximum walking

distance and the working day duration.

In the last decade, researchers have turned their attention to developing exact methods for more restricted TTRP variants (Belenguer et al. 2016; Parragh and Cordeau 2017; Rothenbächer et al. 2018). Belenguer et al. (2016) propose a branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots (STTRPSD). This algorithm is capable of optimally solving instances with up to 50 customers when limiting the number of parking places to 10. Parragh and Cordeau (2017) propose a branch-and-price algorithm to solve the truck and trailer routing problem with time windows (TTRPTW). Their method is capable of solving instances with up to 100 customers. Rothenbächer et al. (2018) propose a branch-price-and-cut algorithm that outperforms the algorithm of Parragh and Cordeau (2017) on the same TTRPTW variant. Their method is capable of finding the optimal solution for 35 additional instances. As opposed to the TTRPTW, customers in the PLRP do not have an associated time window. Therefore, the latter can be expected to be more difficult to solve using branch-price-and-cut.

Another related problem is the doubly open park-and-loop routing problem (DO-PLRP) introduced by Cabrera et al. (2022). This variant of the VRP consists in finding a set of least-cost routes that may start and end at any customer. To solve the DOPLRP, the authors propose a two-phase matheuristic called MSH. This approach was capable of handling instances with up to 3,800 customers. As opposed to the PLRP, the time duration of each walking subtour is bounded. More recently, Le Colleter et al. (2023) defined the park-and-loop routing problem with parking selection (PLRP-PS). The main difference with other problem variants is that the vehicle can only be parked at dedicated parking locations. To solve the PLRP-PS, the authors introduce a small and large neighborhood search metaheuristic (SLNS). The authors use new specific techniques to select parking spots that significantly speed up the algorithm. Their algorithm was capable of providing solutions for instances with up to 400 customers and 352 dedicated parking spots.

The closest problem to the PLRP is the VRP with transportable resources

(VRPTR) introduced by Coindreau et al. (2019). In the VRPTR, a set of workers has to serve a set of customers. The workers can either walk or drive to their next location and are allowed to carpool (i.e., share a vehicle). To solve their VRPTR, the authors use a mixed integer linear program (MILP). Their experiments show that their MILP can only solve instances with up to 18 customers. Thus, they also propose a variable neighborhood search (VNS) algorithm. This method can solve instances with up to 50 customers with a running time limit of 10 hours. The authors also studied a version of their problem in which carpooling is not allowed. The latter perfectly matches the PLRP definition. Le Colleter et al. (2023) reported results benchmarking VNS, MSH, and SLNS on the Coindreau et al. (2019) instances. Their study sets SLNS as the state-of-the-art algorithm since it unveiled eight new best-known solutions. Note, however, that neither of these methods is exact. Moreover, none of those approaches has been assessed with respect to a lower bound. In other words, no optimality gaps have been reported for the solutions they provide.

The contribution of this article is two-fold. From a methodological perspective, we propose a branch-price-and-cut algorithm to solve the PLRP. The key algorithmic component of our method is the pulse algorithm used to solve the pricing problem. The latter extends the procedure introduced in Lozano et al. (2016) to handle the park-and-loop structure of the routes and the inclusion of the subset row inequalities proposed by Jepsen et al. (2008). In addition, we present a set of acceleration strategies tailored to the PLRP. From a computational perspective, we perform extensive experiments on the set of 40 instances introduced by Coindreau et al. (2019), arguably the most widely used testbed for VRPs with park-and-loop structure. Our algorithm is the first to prove optimality for 39 of the instances. To further assess the advantages and limitations of our approach, we also perform experiments on a new challenging set of instances. In addition, we developed an online web application [1] that allows researchers to visualize and download the best-known solutions for the PLRP. They can also upload their solutions for plotting

[1] Available at `https://chairelogistique.hec.ca/en/scientific-data/`

and checking.

This paper is organized as follows. Section 2.2 presents the mathematical formulation. Section 2.3 describes the proposed branch-price-and-cut algorithm. Section 2.4 presents the acceleration strategies that crucially improve the algorithm's performance. Section 2.5 contains the computational experiments. Finally, Section 2.6 presents the conclusions and outlines potential paths for future research.

2.2 Problem formulation

We define a route r as an ordered set of directed arcs starting and ending at the depot. The customers served in the route are represented by the set C_r. Let the subsets $\mathcal{A}_r^d$ and $\mathcal{A}_r^w$ contain the driving and the walking arcs in r respectively. Every arc $(i,j) \in \mathcal{A}_r^w$ must satisfy the condition $\delta_{ij} \leq \theta$. A route r is time-feasible if

$$\sum_{(i,j)\in\mathcal{A}_r^d \cup \mathcal{A}_r^w} \eta_{ij} + \sum_{i\in C_r} s_i \leq \phi. \tag{2.1}$$

Similarly, a route r is walking-feasible if

$$\sum_{(i,j)\in\mathcal{A}_r^w} \delta_{ij} \leq \zeta. \tag{2.2}$$

The cost c_r of a route is equal to the sum of the variable and the fixed cost, that is,

$$c_r = \sum_{(i,j)\in\mathcal{A}_r^d} \delta_{ij} c^v + c^f. \tag{2.3}$$

Let $\mathcal{R}$ be the set of all feasible routes and let a_{ir} be a parameter that takes the value 1 if and only if route $r \in \mathcal{R}$ serves customer $i \in C$. Finally, let x_r be a binary variable equal to 1 if route $r \in \mathcal{R}$ is selected and 0 otherwise. A set covering (SC) formulation for the PLRP can be stated as follows:

$$\min \sum_{r\in\mathcal{R}} x_r c_r \tag{2.4}$$

subject to

$$\sum_{r \in \mathcal{R}} a_{ir} x_r = 1 \qquad \forall i \in C \tag{2.5}$$

$$\sum_{r \in \mathcal{R}} x_r \geq \left\lceil \frac{\sum_{i \in C} s_i}{\phi} \right\rceil \tag{2.6}$$

$$\sum_{r \in \mathcal{R}} x_r \leq k \tag{2.7}$$

$$x_r \in \{0, 1\} \qquad \forall r \in \mathcal{R}. \tag{2.8}$$

The objective function (2.4) minimizes the total cost. Constraints (2.5) ensure that all customers are served exactly once. Constraints (2.6) and (2.7) provide, respectively, a lower and an upper bound on the number of routes used to serve the set of customers. Constraints (2.8) are the domain restrictions. Note that the number of feasible routes $|\mathcal{R}|$ grows exponentially. Thus, solving SC by enumerating all the feasible routes is usually not possible. As an alternative, the SC can be solved using a branch-price-and-cut algorithm which is described next.

2.3 Solution method

In this section, we present an exact branch-price-and-cut algorithm (BPC) to solve the SC model. A BPC algorithm is a branch-and-bound algorithm in which, at each node of the enumeration tree, the linear relaxation of an integer formulation is solved using column generation (CG) and tightened by adding valid inequalities (i.e., cuts). For completeness, Section 2.3.1 describes the column generation algorithm. Section 2.3.2 defines the pricing problem. Section 2.3.3 describes the key strategies used in the pricing problem algorithm. Section 2.3.4 presents the valid inequalities. Finally, Section 2.3.5 describes the branching rules.

2.3.1 Column generation

CG is a solution method that can solve integer programs with a large number of decision variables (i.e., columns). To achieve this goal, two optimization problems are solved iteratively: A *master problem* that considers only a small subset of variables and a *pricing problem* that generates promising variables to be added to the master problem. If, at a given iteration, no variables are added to the master problem, the CG algorithm ends. We refer the interested reader to the book by Desrosiers et al. (2024) and the study by Uchoa et al. (2024) for a review of techniques and applications of column generation.

In our case, the master problem corresponds to the relaxed version of formulation (2.4)-(2.8). The relaxed set covering problem (RSCP) considers only a subset of feasible routes (columns) $\overline{\mathcal{R}} \subseteq \mathcal{R}$ and is obtained by relaxing the integrality constraints on the x_r variables. Let $\pi_i \in \mathbb{R}$, $\sigma \geq 0$, and $\rho \leq 0$ be the dual variables associated with constraints (2.5), (2.6), and (2.7) respectively. The reduced cost of variable x_r is then given by the following expression:

$$\mathfrak{r}_r = c_r - \sum_{i \in C} a_{ir}\pi_i - \sigma - \rho. \tag{2.9}$$

Having $\min_{r \in \overline{\mathcal{R}}} \{\mathfrak{r}_r\} \geq 0$ ensures that the RSCP is solved optimally. If there exists a route r with $\mathfrak{r}_r < 0$, we add the corresponding variable x_r to the subset $\overline{\mathcal{R}}$. Therefore, the goal after solving the RSCP is to identify a route with negative reduced cost. This problem is referred to as the pricing problem and is the topic of the next section.

2.3.2 Pricing problem

Let $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ be a directed multigraph, henceforth referred to as the *modified network*, where $\mathcal{V}'$ is the set of nodes including a start depot $\underline{0}$ and an end depot $\overline{0}$. Thus, $\mathcal{V}' = C \cup \{\underline{0}, \overline{0}\}$ and $\mathcal{A}' = \mathcal{A}^1 \cup \mathcal{A}^2 \cup \mathcal{A}^3 \cup \mathcal{A}^4 \cup \mathcal{A}^5$, where

- $\mathcal{A}^1 = \left\{(i,j)^d | i \in C \cup \left\{\underline{0}\right\}, j \in C\right\}$ is the set of driving arcs arriving to any customer,

- $\mathcal{A}^2 = \left\{(i,j)^w | i \in C \cup \left\{\underline{0}\right\}, j \in C | \delta_{ij} \leq \theta\right\}$ is the set of walking arcs arriving to any customer,

- $\mathcal{A}^3 = \left\{(i,\overline{0})^d | i \in C\right\}$ is the set of driving arcs going from any customer to the end depot,

- $\mathcal{A}^4 = \left\{(i,\underline{0})^w | i \in C | \delta_{i\underline{0}} \leq \theta\right\}$ is the set of walking arcs going from any customer to the start depot, and

- $\mathcal{A}^5 = \left\{(\underline{0},\overline{0})^w\right\}$ is a fictitious arc connecting the start and end depots.

The attributes of the arc connecting $\underline{0}$ and $\overline{0}$ are set to zero. Note that arcs $\mathcal{A}^4$ are necessary in order to allow workers to start walking subtours from the depot. Figure 2.2 shows how the routes in Figure 2.1b are mapped to the modified network.

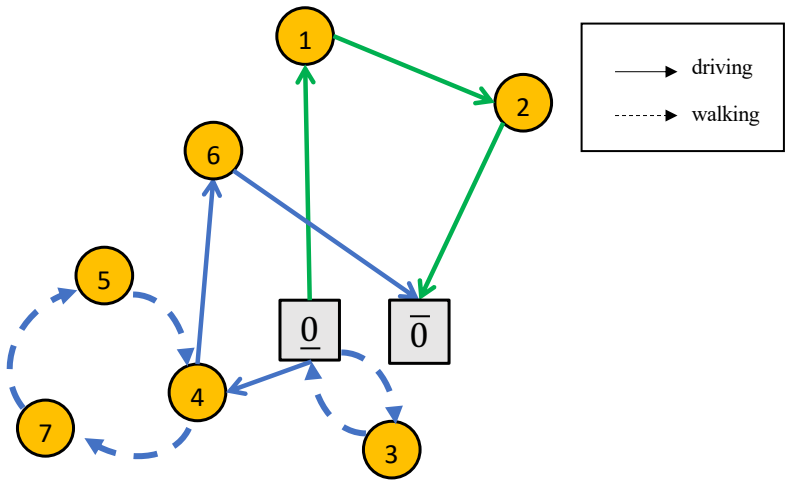


Figure 2.2: Route representation on the modified network.

As mentioned previously, the objective of the pricing problem is to find routes from $\underline{0}$ to $\overline{0}$ with a negative reduced cost. With this in mind, we must define the reduced cost of each arc in $\mathcal{G}'$. The reduced cost of an arc $(i,j)^m \in \mathcal{A}'$ is defined as:

$$r_{ij} = \begin{cases} \delta_{ij} c^v - \pi_j, & (i,j)^m \in \mathcal{A}^1; \\ -\pi_j, & (i,j)^m \in \mathcal{A}^2; \\ \delta_{ij} c^v, & (i,j)^m \in \mathcal{A}^3; \\ 0, & (i,j)^m \in \mathcal{A}^4; \\ 0, & (i,j)^m \in \mathcal{A}^5. \end{cases} \tag{2.10}$$

In terms of network flows, the start depot offers one unit of flow that is demanded by the end depot. Thus, the formulation of the pricing problem partially follows that of the shortest path problem in which the weight of the arcs corresponds to their reduced cost. Nevertheless, in the context of the PLRP, a node in graph $\mathcal{G}'$ can only be visited once (unless the node is used as a parking spot), which means that paths are pseudo-elementary. Moreover, paths (i.e., routes) are constrained by two resources: the time ϕ and the walking distance ζ. Hence, the pricing problem corresponds to solving a variant of the elementary shortest path problem with resource constraints (ESPPRC) in which routes have a park-and-loop structure. We will refer to our specific pricing problem as the elementary shortest path problem with resource constraints and park-and-loop (ESPPRC-PL).

The ESPPRC itself is NP-hard. For this reason, considerable research effort has been devoted to designing algorithms to solve versions of the problem in which the elementarity constraints are partially (or completely) relaxed. Even though the quality of the bounds provided by the MP may be significantly reduced, solving these relaxations is usually faster. The interested reader is referred to Costa et al. (2019b) for a detailed review of ESPPRC relaxations. The resulting shortest path problem with resource constraints (SPPRC) is often solved using dynamic programming algorithms. To put it simply, these algorithms are based on the idea of extending labels following a breadth-first search paradigm. Each label represents a partial path and stores the consumption of resources. In order to avoid a complete enumeration of the partial paths in the network, labels are discarded using problem-

specific dominance rules. These algorithms can be very fast for small and medium-sized networks. Nevertheless, the number of labels to store can be extremely large. Thus, they may fail to scale to large networks.

Another stream of research has focused on solving the ESPPRC without explicitly applying any relaxations. As a result, the bounds provided by the MP are typically tighter (stronger), which potentially leads to the exploration of smaller branch-and-price trees. One of the algorithms that is commonly used in this setting is the pulse algorithm (PA) proposed by Lozano et al. (2016). The PA is based on a recursive depth-first search that crucially combines multiple pruning strategies that discard partial paths. A vital feature of the PA is that it does not rely on asserting dominance between labels. More crucially, labels are never stored at nodes, which significantly decreases the memory requirements of the algorithm. Another interesting feature of the PA is its flexibility. In recent years, it has been adapted (with mild changes) to solve the orienteering problem with time windows (Duque et al. 2014), the bi-objective shortest path problem (Duque et al. 2015), the weight-constrained shortest path problem with replenishment (Bolívar et al. 2014), the shortest α-reliable path problem (Corredor-Montenegro et al. 2021), and, more recently, the least expected travel-time path problem (Yamín et al. 2022). The PA has been also used as a component in algorithms to solve other hard combinatorial optimization problems (Arslan et al. 2018; Lozano and Smith 2017; Montoya et al. 2016; Restrepo et al. 2012; Schrotenboer et al. 2019).

Based on the above observations, we decided to adapt the PA to solve the ESPPRC-PL. Following the same intuition of the original PA, our algorithm propagates pulses through the network from a start node (i.e., the start depot) $\underline{0}$ to an end node (i.e., the end depot) $\overline{0}$. While traversing the network node by node, the pulse builds a partial path $\mathcal{P}$ that includes all the nodes already visited. Additionally, the pulse contains information on the attributes associated with the path, such as the cumulative reduced cost or the resource consumption. Whenever a pulse reaches the end node $\overline{0}$ it contains all the information of a feasible path $\mathcal{P}$ from $\underline{0}$ to $\overline{0}$. If

the path has a negative reduced cost, it can be added to the subset $\overline{\mathcal{R}}$ and the best solution can be updated.

The PA ensures that the optimal path $\mathcal{P}^*$ is always found by implicitly enumerating all paths from $\underline{0}$ to $\overline{0}$. However, one can easily truncate the PA to accelerate the search by solving the problem heuristically, as it is discussed in Section 2.4.2. To prevent the PA from explicitly enumerating all possible paths, the algorithm uses a set of pruning strategies. These strategies allow the PA to stop (prune) the propagation of a partial path as soon as there is enough evidence that the partial path will not improve the current best solution or that it will lead to an infeasible solution. Note that stopping a partial path from propagating allows for discarding a large number of paths, as it discards all the paths that begin with it. Thus, the earlier a partial path is stopped, the better. This idea is shared with other algorithms, like branch-and-bound, where an implicit enumeration is performed. Similarly, the strength of the pulse algorithm depends on the pruning strategies. In what follows the terms *stopping* or *pruning* a path are used interchangeably.

Algorithm 1 presents the main logic of the pulse algorithm. Line 1 initializes the partial path. Lines 2 to 4 initialize the reduced cost and the cumulative resource consumption. Line 5 runs the bounding procedure given the bound step size Δ and the bounding time limits $[\overline{t}, \underline{t}]$. Line 6 extends a pulse at the start node. Finally, line 7 returns the optimal path. Note that in the case where reaching the end node $\overline{0}$ is not possible due to the resource constraints (i.e., the pricing problem is infeasible), the optimal path ends up empty.

Algorithm 2 shows the recursive procedure `pulse`, where $\Gamma_w^+(i)$ corresponds to the set of arcs leaving node i on foot and $\Gamma_d^+(i)$ by driving. Lines 1 to 4 use the pruning strategies, namely, infeasibility, bounds, rollback, and path completion, to try to prune a partial path. If the pulse is not pruned, line 5 adds the node to the partial path. From lines 6 to 10, the algorithm recursively propagates the pulse by driving to node j. From lines 11 to 17, the algorithm recursively propagates the pulse using Algorithm 3 by walking to node j, thus parking the vehicle at node i.

Algorithm 1 `pulseSearch` function

Require: $\mathcal{G}'$, directed multi graph; ϕ, duration limit; ζ, walking distance limit; $\underline{0}$, start node; $\overline{0}$, end node; Δ, bound step size; $[\bar{t}, \underline{t}]$, bounding time limits.

Ensure: $\mathcal{P}^*$, optimal path.

1: $\mathcal{P}^* \leftarrow \emptyset$
2: $r(\mathcal{P}) \leftarrow 0$
3: $t(\mathcal{P}) \leftarrow 0$
4: $w(\mathcal{P}) \leftarrow 0$
5: `bound`$(\mathcal{G}', \Delta, [\bar{t}, \underline{t}])$ ▷ see *§2.3.3*
6: `pulse`$(\underline{0}, r(\mathcal{P}), t(\mathcal{P}), w(\mathcal{P}), \mathcal{P})$ ▷ see *Algorithm 2*
7: **return** $\mathcal{P}^*$

Algorithm 2 `pulse` function

Require: i, current node; r, cumulative reduced cost; t, cumulative time; w, cumulative walking distance; $\mathcal{P}$, partial path.

1: **if** ¬`feasibility`$(i, t(\mathcal{P}), w(\mathcal{P}))$ **then** ▷ see *§2.3.3*
2: **if** ¬`bounds`$(i, r(\mathcal{P}), t(\mathcal{P}))$ **then** ▷ see *§2.3.3*
3: **if** ¬`rollback`$(i, r(\mathcal{P}), t(\mathcal{P}), w(\mathcal{P}), \mathcal{P})$ **then** ▷ see *§2.3.3*
4: **if** ¬`complete_path`$(i, r(\mathcal{P}), t(\mathcal{P}), w(\mathcal{P}), \mathcal{P})$ **then** ▷ see *§2.3.3*
5: $\mathcal{P}' \leftarrow \mathcal{P} \cup \{i\}$
6: **for** $j \in \Gamma_d^+(i)$ **do**
7: $r(\mathcal{P}') \leftarrow r(\mathcal{P}) + r_{ij}$
8: $t(\mathcal{P}') \leftarrow t(\mathcal{P}) + \eta_{ij} + s_j$
9: `pulse`$(j, r(\mathcal{P}'), t(\mathcal{P}'), w(\mathcal{P}), \mathcal{P}')$
10: **end for**
11: **for** $j \in \Gamma_w^+(i)$ **do**
12: $r(\mathcal{P}') \leftarrow r(\mathcal{P}) + r_{ij}$
13: $t(\mathcal{P}') \leftarrow t(\mathcal{P}) + \eta_{ij} + s_j$
14: $w(\mathcal{P}') \leftarrow w(\mathcal{P}) + \delta_{ij}$
15: $ps \leftarrow i$
16: `pulse_parked`$(j, r(\mathcal{P}'), t(\mathcal{P}'), w(\mathcal{P}'), ps, \mathcal{P}')$
17: **end for**
18: **end if**
19: **end if**
20: **end if**
21: **end if**
22: **return** `void`

Algorithm 3 shows the recursive procedure `pulse_parked`. Lines 1 to 3 try to prune the partial path using the infeasibility, bounds, and rollback pruning strategies. If the pulse is not pruned, line 4 adds the node to the partial path. From lines

5 to 22, the algorithm recursively propagates the pulse by walking to node j. At line 9, we check if the path is returning to the parking spot. If so, at line 14 the algorithm checks if it is possible to prune the partial path by using the subtour fixing strategy. If the partial path is not pruned, the algorithm recursively propagates the pulse using Algorithm 2.

Algorithm 3 `pulse_parked` function

Require: i, current node; r, cumulative reduced cost; t, cumulative time; w, cumulative walking distance; ps, parking spot; $\mathcal{P}$, partial path.

1: **if** $\neg$`feasibility`$(i, t(\mathcal{P}), w(\mathcal{P}))$ **then** ▷ see *§2.3.3*
2: **if** $\neg$`bounds`$(i, r(\mathcal{P}), t(\mathcal{P}))$ **then** ▷ see *§2.3.3*
3: **if** $\neg$`rollback`$(i, r(\mathcal{P}), t(\mathcal{P}), w(\mathcal{P}), \mathcal{P})$ **then** ▷ see *§2.3.3*
4: $\mathcal{P}' \leftarrow \mathcal{P} \cup \{i\}$
5: **for** $j \in \Gamma_w^+(i)$ **do**
6: $r(\mathcal{P}') \leftarrow r(\mathcal{P}) + r_{ij}$
7: $t(\mathcal{P}') \leftarrow t(\mathcal{P}) + \eta_{ij} + s_j$
8: $w(\mathcal{P}') \leftarrow w(\mathcal{P}) + \delta_{ij}$
9: **if** $ps = j$ **then**
10: $t(\mathcal{P}') \leftarrow t(\mathcal{P}') - s_j$
11: **if** $j \in C$ **then**
12: $r(\mathcal{P}') \leftarrow r(\mathcal{P}) + \pi_j$
13: **end if**
14: **if** $\neg$`subtour_fixing`$(j, t(\mathcal{P}), \mathcal{P})$ **then** ▷ see *§2.3.3*
15: `pulse`$(j, r(\mathcal{P}'), t(\mathcal{P}'), w(\mathcal{P}'), \mathcal{P}')$
16: **end if**
17: **else if** $ps \neq j$ **then**
18: `pulse_parked`$(j, r(\mathcal{P}'), t(\mathcal{P}'), w(\mathcal{P}'), ps, \mathcal{P}')$
19: **end if**
20: **end for**
21: **end if**
22: **end if**
23: **end if**
24: **return** `void`

The following section provides further detail regarding the pruning strategies used by the PA.

2.3.3 Pruning strategies

In Sections 2.3.3, 2.3.3, and 2.3.3 we describe the adaptation of the original PA pruning strategies proposed by Lozano et al. (2016), namely, infeasibility, bounds, and rollback pruning. In Section 2.3.3 we present the path completion strategy (line 4 in Algorithm 2) adapted from Cabrera et al. (2020) which was used to solve the constrained shortest path problem (CSP). Finally, in Section 2.3.3 we describe a new pruning strategy for the PA specifically tailored for our pricing problem called subtour fixing (line 16 in Algorithm 3).

Infeasibility pruning

The intuition of this pruning strategy is to stop a pulse as soon as it becomes evident that it will not be able to reach the end node $\overline{0}$ while meeting the resource constraints. Thus, we can safely stop a partial path $\mathcal{P}$ from propagating if any of the following conditions holds:

- $t(\mathcal{P}) > \phi$;
- $w(\mathcal{P}) > \zeta$.

Discarding infeasible partial paths is often used as a key strategy to improve the performance of labeling algorithms. Note that this strategy could easily be extended to include other route constraints such as the presence of time windows at customer locations or a time limit on each walking subtour.

Bounds pruning

Similar to the infeasibility pruning strategy, we can stop a pulse from propagating if there is enough information to prove that the current partial path will not lead to improving the best solution found so far. More specifically, if there is evidence that the partial path will not be able to decrease the best objective function $r(\mathcal{P}^*)$ we can stop the partial path from propagating. With this purpose, we use a bounding

scheme that computes lower bounds $\underline{r}(i, t(\mathcal{P}))$ for every node $i \in \mathcal{G}'$ and for a set of possible values of time resource consumption $t(\mathcal{P})$. More specifically, these bounds contain the minimum reduced cost from any node i to the end node $\overline{0}$ given a partial resource consumption $t(\mathcal{P})$.

To compute the lower bounds, we solve an ESPPRC-PL from every node $i \in \mathcal{G}'$ to $\overline{0}$ given a time consumption of $t(\mathcal{P}) = \bar{t} - \Delta$. These problems are overly-constrained as the pulse only has Δ units of time available to reach the end node $\overline{0}$. Accordingly, the `pulse` algorithm can easily solve these problems to optimality. Each of the solutions found is a valid lower bound on the minimum reduced cost that can be obtained from node i given a time consumption $t(\mathcal{P}) \geq \bar{t} - \Delta$. After finding these bounds, we proceed to solve an ESPPRC-PL from every node $i \in \mathcal{G}'$ to $\overline{0}$ given a time consumption of $t(\mathcal{P}) = \bar{t} - 2\Delta$. Although the resulting problems are less constrained, we already have vital information for partial paths with a time consumption between $[\bar{t} - \Delta, \bar{t}]$. We continue with this procedure, solving ESPPRC-PL problems with $\left\{\bar{t} - 3\Delta, \bar{t} - 4\Delta, ..., \underline{t}\right\}$. Note that at the end of this procedure, we have a lower bound for every node and every discrete time step between $\underline{t}$ and $\bar{t}$. With this information, we can prune a partial path $\mathcal{P}$ if $r(\mathcal{P}) + \underline{r}(i, t(\mathcal{P})) \geq r(\mathcal{P}^*)$.

For further details regarding this strategy, the reader is referred to Lozano et al. (2016). Note that in this section we describe the bounding scheme using only the time consumption $t(\mathcal{P})$. In preliminary experiments, we considered computing additional bounds with respect to the walking distance $w(\mathcal{P})$. However, this did not lead to a major improvement in performance.

Rollback pruning

The choice between a depth-first search (DFS) and a breadth-first search (BFS) strategy has been widely studied in the literature as it affects the performance of every labeling setting/correcting algorithm. Although it is possible to affect the behavior of the pulse algorithm to make a BFS exploration through the usage of *pulse queues* as presented in Cabrera et al. (2020), in this article we consider a

version of the pulse algorithm that follows a pure DFS strategy.

In some cases, this behaviour could lead to exploring vast unpromising regions of the search space, before backtracking to correct poor decisions made earlier. To overcome this problem, the rollback pruning strategy re-evaluates the last choice made. More specifically, consider a partial path $\mathcal{P}_{\underline{0},i}$ from $\underline{0}$ to i that is extended to node l and then reaches node j, namely, $\mathcal{P}_{\underline{0},j} = \mathcal{P}_{\underline{0},i} \cup l \cup j$. Once the partial path $\mathcal{P}_{\underline{0},j}$ reaches node j, we compare it with the partial path which skips node l, that is, $\mathcal{P}'_{\underline{0},j} = \mathcal{P}_{\underline{0},i} \cup j$. If $r(\mathcal{P}'_{\underline{0},j}) \leq r(\mathcal{P}_{\underline{0},j})$ and $t(\mathcal{P}'_{\underline{0},j}) \leq t(\mathcal{P}_{\underline{0},j})$ we can prune path $\mathcal{P}_{\underline{0},j}$ as it is dominated by path $\mathcal{P}'_{\underline{0},j}$. As pointed out by Lozano et al. (2016), this strategy is based on simple arithmetic calculations for $r(\mathcal{P}'_{\underline{0},j})$ and $t(\mathcal{P}'_{\underline{0},j})$ and does not rely on storing any kind of labels.

Path completion

One of the main drawbacks of the PA, as it was presented by Lozano et al. (2016), is that the best path $\mathcal{P}^*$ is only updated when the end node $\overline{0}$ is reached. Thus, the main purpose of the path completion strategy is to update the best path (primal bound) at intermediate nodes in the network.

To do so, we take advantage of the information computed in the bounding procedure presented in Section 2.3.3. Formally, let us consider a partial path $\mathcal{P}_{\underline{0},i}$ arriving to node i. The path completion strategy adds the minimum reduced cost path $\mathcal{P}^{rt}_{i,\overline{0}}$ given a time consumption $t(\mathcal{P}_{\underline{0},i})$ to the partial path $\mathcal{P}_{\underline{0},i}$, that is, $\mathcal{P}_{\underline{0},\overline{0}} = \mathcal{P}_{\underline{0},i} \cup \mathcal{P}^{rt}_{i,\overline{0}}$. If the completed path is feasible and the reduced cost is lower than the current primal bound, we update the incumbent solution accordingly. Furthermore, we can stop the incoming partial path $\mathcal{P}_{\underline{0},i}$ from propagating, because (by construction) we know that the complete path is already the minimum reduced cost path stemming from this partial path given the current time consumption. This procedure is used in Line 4 of Algorithm 2 and is adapted from the path completion strategy proposed by Cabrera et al. (2020) for the constrained shortest path problem.

Subtour fixing

Consider a partial path $\mathcal{P}_{\underline{0},j}$ from $\underline{0}$ to j in which j is used as a parking spot. Moreover, consider a walking subtour s visiting a subset C_s^j of three or more customers $C_s^j = \left\{i_1, i_2, ..., i_{|C_s^j|}\right\}$ and stemming from node j. The number of possible walking subtours visiting all the customers in C_s^j and using j as a parking spot can be calculated as $|C_s^j|!$. Thus, during the recursive search, the algorithm can visit j to retrieve the vehicle following several different sequences in which all the customers in C_s^j are visited. The total reduced cost associated with each of these sequences is the same (i.e., $\sum_{i \in C_s^j} -\pi_i$), while the total time may differ. Note that a slower walking subtour may result in the impossibility of visiting other customers using the partial path $\mathcal{P}_{\underline{0},j}$.

Once again, as the algorithm follows a pure DFS strategy, is possible that many partial paths will be explored before backtracking and correcting the sequence followed for visiting the customers in C_s^j. For this reason, every time a partial path completes a walking subtour s we check if it is possible to stop that path from propagating. In practice, we follow two steps. First, we check if it is the first time (since the BPC algorithm started) that the customers C_s^j are visited by a walking subtour stemming from j. If so, we solve a traveling salesman problem minimizing the total walking time and store the value $t^*(C_s^j)$ in memory. If not, we retrieve the value stored previously. Second, we compare the current time of the subtour $t(s)$ with the best time $t^*(C_s^j)$. If the total time $t(s)$ of the subtour s currently included in the partial path $\mathcal{P}_{\underline{0},j}$ is larger than the total time of the optimal TSP (i.e., $t(s) > t^*(C_s^j)$) we can prune the path and thus avoid exploring paths that use an inefficient walking subtour.

2.3.4 Valid inequalities

The optimal solution of the RSCP can be fractional. In that case, before applying branching decisions we first try to improve (lift) the lower bound. To do so,

we draw upon valid inequalities (cuts). Particularly, we include the subset row inequalities proposed by Jepsen et al. (2008) for subsets of three customers. These inequalities have been used in different applications as the multi-depot vehicle routing problem (Contardo and Martinelli 2014), the vehicle routing problem with time windows (Costa et al. 2019b), and the two-echelon capacitated vehicle routing problem (Marques et al. 2020). The subset row inequalities with $|\mathcal{S}| = 3$ are defined as

$$\sum_{r \in \mathcal{R}} \left\lfloor 1/2 \sum_{i \in \mathcal{S}} a_{ir} \right\rfloor x_r \leq 1, \qquad \forall \mathcal{S} \subseteq \mathcal{C}. \tag{2.11}$$

Each of these inequalities has an associated dual variable $\beta_{\mathcal{S}} \leq 0$. These inequalities ensure that for a given subset $\mathcal{S} \subseteq \mathcal{C}$ the number of routes serving two or more customers is less or equal to 1.

To separate these inequalities we enumerate all customer triplets and check if the inequality is violated in the current optimal solution. It has been observed by multiple researchers that even if subset row inequalities tend to have a positive impact on the quality of the lower bound, they significantly increase the complexity of solving the pricing problem. Thus, in line with Jepsen et al. (2008), we allow our BPC algorithm to add up to φ at each iteration with a minimum violation of ε. Inequalities with greater violations are given priority. In our BPC, we set φ to 5 and ε to 0.1.

Note that adding these inequalities modifies the definition of the reduced cost of a route. More specifically, if we denote $\mathcal{S}$ as the subset of triplets of customers for which the subset row inequality has been generated and added to the master problem, then the reduced cost of a route is defined as

$$\mathfrak{r}_r = c_r - \sum_{i \in \mathcal{C}} a_{ir} \pi_i - \sigma - \rho - \sum_{\mathcal{S} \in \mathcal{S}} \beta_{\mathcal{S}} \left\lfloor \frac{1}{2} \sum_{i \in \mathcal{S}} a_{ir} \right\rfloor. \tag{2.12}$$

Adding a subset row inequality implies that a penalty of $\beta_{\mathcal{S}}$ must be paid if two or more customers of the corresponding triplet are served by the route. To account for this term using the PA, we add a new resource for each subset in $\mathcal{S}$ that stores the number of times that a customer in the subset has been visited. If, while extending a pulse this value reaches a value of 2 we subtract the $\beta_{\mathcal{S}}$ from the cumulative reduced cost. The reader should note that all the pruning strategies outlined in Section 2.3.3 can still be used without any changes. This is an advantage compared to algorithms that rely on assessing dominance between two partial paths (labels) in which extending the dominance criteria is required.

In addition, to strengthen the linear relaxation of the SC formulation, we lifted constraint (2.6) using the value of the objective function of the minimum spanning tree on the graph $\mathcal{G}$, as follows:

$$\left\lceil \frac{\sum_{i \in C} s_i + MST(\mathcal{G})}{\phi} \right\rceil \leq \sum_{r \in \mathcal{R}} x_r. \tag{2.13}$$

Constraint (2.13) ensures that the number of routes in the solution of the RSCP is at least the minimum number of routes needed to serve all the customers. Lifting this constraint does not have any impact on the pricing problem structure.

2.3.5 Branching rules

Adding the inequalities outlined in Section 2.3.4 does not guarantee that the optimal solution of the RSCP will be integral. In the case in which the optimal solution of the RSCP is still fractional, we resort to branching on the arc flow variables. To do so, we define a^m_{ijr} as the number of times an arc $(i,j)^m \in \mathcal{A}$ appears in route r. Then, for each arc $(i,j)^m \in \mathcal{A}$ it is possible to compute the number of times it appears in a solution as

$$b^m_{ij} = \sum_{r \in \overline{\mathcal{R}}} a^m_{ijr}. \tag{2.14}$$

We select for branching the arc for which the value of b_{ij}^m is closest to 0.5. In the case of a tie, we prioritize driving arcs.

Branching on an arc implies creating two child nodes for the branch-and-price tree: one child in which the arc is forbidden (the value is set to zero) and one child in which the arc is fixed (the value is set to one). These conditions are enforced locally in the pricing problem by modifying the graph $\mathcal{G}'$. If an arc must be forbidden, the arc is simply removed from graph $\mathcal{G}'$. However, fixing an arc is not as straightforward as it depends on the transportation mode. If the arc (i, j) is a driving arc, we remove all the driving arcs starting from node i and ending at any node different than j. Moreover, we remove all the driving arcs ending at node j that start at any node different than i. In addition, we remove the walking arcs $(i, j)^w$ and $(j, i)^w$. If the arc (i, j) is a walking arc, we remove all the walking arcs starting from node i and ending at any node different than j. We also remove all the walking arcs ending at node j that start at any node different than i. In addition, we remove the driving arcs $(i, j)^d$ and $(j, i)^d$. In order to select the next node to branch on we use best bound search.

2.4 Acceleration strategies

We now describe several ideas that we use to speed up our BPC algorithm.

2.4.1 Dual stabilization

Usually, CG-based algorithms suffer from slow convergence, a phenomenon called the *tailing-off effect* (Desrosiers et al. 2024). An important technique for alleviating this issue is to implement a dual stabilization method. In this work, we implemented the α-schedule procedure presented by Pessoa et al. (2013). This procedure aims to correct the values of the dual variables used to solve the pricing problem based on previous dual solutions. Algorithm 4 shows the pseudocode of the procedure. Line

1 initializes the value of l. Line 2 initializes the value of the dual variables. Line 3 updates the smoothing factor. Line 4 computes the value of the dual variables that will be used for solving the pricing problem. Line 5 updates the value of l. Line 6 calls the pricing problem solved with the procedure described in Section 2.3.2. Solving the pricing problem with the modified dual variables may result in the impossibility to find a column with a negative reduced cost, even if such a column exists. This phenomenon is called *mis-pricing*. If *mis-pricing* occurs the algorithm returns to line 3. Otherwise, the number of iterations is updated, and the master problem is solved once again.

Algorithm 4 α-scheduling function

Require: α, smoothing factor.

1: $l \leftarrow 1$
2: $\pi^0 \leftarrow \pi^{in}$
3: $\overline{\alpha} \leftarrow [1 - l(1-\alpha)]^+$
4: $\pi^{sep} = \overline{\alpha}\pi^0 + (1-\overline{\alpha})\pi^{out}$
5: $l \leftarrow l + 1$
6: Call the pricing problem with π^{sep}
7: **if** Mis-pricing occurs **then**
8: Go to step 3
9: **else**
10: $t \leftarrow t + 1$
11: Solve the master problem
12: Go to step 1
13: **end if**

2.4.2 Heuristic pulse algorithm

The pricing problem does not have to be solved to optimality at every iteration of the CG. Thus, it is a common practice to design heuristics to quickly find promising solutions in the first iterations of the CG (Desaulniers et al. 2005). In our case, we can easily truncate the PA to solve the pricing problem heuristically by imposing a stopping criterion. More specifically, we can heuristically stop the PA from propagating more pulses if the number of paths found with negative reduced cost reaches

Υ. Moreover, we can impose a time limit Λ. Then, if the CPU time for solving the pricing problem reaches Λ and the PA has already found a promising path, we stop the PA.

Furthermore, note that by allowing the PA to find paths with a park-and-loop structure, the complexity of solving the pricing problem increases heavily. However, it is possible that in some iterations of the CG, paths without subtours may have a negative reduced cost. Accordingly, we adopt a leveled pricing strategy in which we first run the PA without considering the walking arcs. Only if the algorithm was not able to find promising paths do we proceed to run the PA by allowing walking subtours.

2.4.3 Initialization

It is well known that the performance of a CG algorithm is affected by the initial set of columns. Usually, including a high-quality set of initial columns can help the algorithm perform a better estimation of the dual variables associated with the RSCP constraints. Although one could initialize the pool of columns using $|C|$ routes visiting one customer, in our implementation we use the output of the sampling phase of the MSH matheuristic proposed by Cabrera et al. (2022). For the sake of completeness, we briefly describe the procedure here.

Algorithm 1 presents the main logic of the sampling phase of MSH. Line 1 initializes the set of initial routes $\overline{\mathcal{R}}$. Line 2 initializes the iteration number. From lines 4 to 13, the algorithm populates $\overline{\mathcal{R}}$ using a set of TSP heuristics $\mathcal{H}$ and the splitting procedure $split\langle\cdot,\cdot\rangle$. Line 4 randomly selects a TSP heuristic h from $\mathcal{H}$. Line 5 generates a giant route τ^t visiting all customers using h. Line 6 generates a solution, denoted as s^t. Line 7 joins the routes in solution s^t to set $\overline{\mathcal{R}}$. Lines 8-12 update the incumbent solution. Line 15 returns the set of initial routes that will be used by our BPC.

Algorithm 5 `MSH sampling` function

Require: G, graph; $\mathcal{H}$, heuristic set; T, iteration limit; Q, time limit.
Ensure: initial solution $\overline{\mathcal{R}}$

1: $\overline{\mathcal{R}} \leftarrow \emptyset$
2: $t \leftarrow 1$
3: **while** $t < T \wedge samplingTime < Q$ **do**
4: $\quad h \leftarrow \mathcal{H}$
5: $\quad \tau^t \leftarrow h(\mathsf{G})$
6: $\quad s^t \leftarrow split\langle \mathcal{G}, \tau^t \rangle$
7: $\quad \overline{\mathcal{R}} \leftarrow \overline{\mathcal{R}} \cup s^t$
8: $\quad$ **if** t $= 1$ **then**
9: $\quad\quad s^* \leftarrow s^t$
10: $\quad$ **else if** $f(s^t) < f(s^*)$ **then**
11: $\quad\quad s^* \leftarrow s^t$
12: $\quad$ **end if**
13: $\quad t \leftarrow t + 1$
14: **end while**
15: **return** $\overline{\mathcal{R}}$

The key algorithmic component of the sampling phase of MSH is the $split\langle \cdot, \cdot \rangle$ procedure used to extract a solution s^t from the giant TSP-like tour in Line 6. The split procedure follows two steps. In the first step, it constructs a directed acyclic graph defined by a set of nodes $\mathsf{N} = (v_0, v_1, ..., v_i, ..., v_n)$ and the set of arcs A. Node v_0 is a dummy node, while nodes numbered 1 to n represent the customer in the i-th position of the giant tour τ^t. Each arc $(i, j) \in \mathsf{A}$ represents a feasible route $r(v_{i+1}, v_j)$ visiting customers $(v_{i+1}, ..., v_j)$. To evaluate if an arc should be added to G (line 12) it solves a subproblem that can be seen as a multi-resource version of the single truck and trailer routing problem with satellite depots (STTRPSD) proposed in Villegas et al. (2010). The solution to this subproblem yields a route with one or more walking subtours. In the second step, the split procedure finds the shortest path from v_0 to v_n in G. The set of arcs (i.e., routes) along the shortest path corresponds to a feasible solution s^t.

In our BPC algorithm, we use the default parameters of the MSH. Namely, the number of iterations is set to 2,500, and the time limit is set to 60 seconds. For further detail regarding the MSH, the reader is referred to Cabrera et al. (2022).

2.5 Computational experiments

In this section, we present the computational experiments that we performed on a set of standard instances from the literature. Our goal is to analyze the performance of the proposed BPC algorithm and its main components. In addition, we describe a web application that can be used to check the quality of the solutions found by any researcher working on the PLRP or related variants. The BPC algorithm was implemented in Java using the jORLib[2] library and compiled using Java 1.8.0_331. The experiments were performed on an Intel core i7 @2.30 GHz Quad-Core processor with 12GB of RAM. We used CPLEX 20.1 to solve the SC formulation and the RSCP. Due to the randomness induced by the initialization procedure described in Section 2.4.3, for every experiment, we ran five replicates. On each replicate, we used a different value from the set $\{1, 2, 3, 4, 5\}$ to seed Java's pseudo-random number generator. This way we ensure consistency across the initial solutions used by each algorithm and validate the consistency of the proposed BPC. We also set a time limit of 2 hours for every run.

2.5.1 Test instances

To assess the efficiency and effectiveness of the BPC algorithm, we use the set of instances proposed by Coindreau et al. (2019) for the VRPTR without carpooling. Each instance considers a number of customers n in the set $\{20, 30, 40, 50\}$ located inside a square grid of 10 km by 10 km. The depot is located at the center of the grid. The distance between nodes (customers and depot) is the Euclidean distance. In addition, to compute driving and walking times they consider a driving speed of 30 km/h and a walking speed of 4 km/h. The customer service times range from 20 to 35 minutes. The maximum daily walking distance for each worker is 5 km and the day duration is 7 hours (i.e., 420 minutes). It is worth recalling that any customer location can be used as a parking spot. For each instance, Coindreau et al. (2019)

[2]The latest version of jORLib can be downloaded at: http://coin-or.github.io/jorlib/.

fixed the number of available workers as the number of routes in the corresponding VRP solution. The fixed cost is set to 0\$ and the variable cost is set to 1\$/km. The objective is to minimize the total cost. An instance is referred to as "`n_A_i`", where `n` stands for the number of customers, `A` represents the size of the used time window (i.e., all day), and `i` denotes the instance unique identifier. A total of 10 instances are considered for each instance size (i.e., number of customers). After fine tuning, we set the bound step size in the PA (Δ) to 15. The bounding time limits $[\underline{t}, \bar{t}]$ are set to $[120, 420]$. In addition, the maximum number of paths Υ is set to 10 and the time limit Λ to 15 seconds. Finally, the parameter α used to stabilize the values of the dual variables is set to 0.8.

2.5.2 Assessing the BPC performance

In this section, we analyze the performance of the proposed BPC and compare it to that of the state-of-the-art algorithms for the PLRP, namely, the VNS introduced by Coindreau et al. (2019), the MSH designed by Cabrera et al. (2022), and the SLNS developed by Le Colleter et al. (2023).

Table 2.1 compares the performance of the proposed branch-price-and-cut algorithm in the best performing replicate against the benchmark algorithms (metaheuristics) in terms of solution quality. Each row corresponds to an instance size. Columns 2, 5, 8, and 11 show the number of best-known solutions (BKSs) found by each algorithm. Columns 3, 6, 9, and 12 report the average gap with the best-known solution. Columns 4, 7, 10, and 13 show the maximum gap with the best-known solution.

Table 2.1: Solution quality on the Coindreau et al. (2019) instances.

$\lvert C \rvert$	VNS			MSH			SLNS			BPC		
	BKSs	Avg. Δ	Max. Δ	BKSs	Avg. Δ	Max. Δ	BKSs	Avg. Δ	Max. Δ	BKSs	Avg. Δ	Max. Δ
20	1/10	4.09%	8.87%	10/10	0.00%	0.00%	10/10	0.00%	0.00%	10/10	0.00%	0.00%
30	0/10	3.13%	5.89%	7/10	0.01%	0.04%	8/10	0.00%	0.04%	10/10	0.00%	0.00%
40	3/10	1.23%	5.31%	6/10	0.15%	0.69%	7/10	0.22%	1.39%	10/10	0.00%	0.00%
50	2/10	2.12%	6.11%	2/10	0.91%	2.60%	4/10	0.14%	0.54%	10/10	0.00%	0.00%
Total/Avg.	6/40	2.64%	6.54%	25/40	0.27%	0.83%	29/40	0.09%	0.49%	40/40	0.00%	0.00%

As the results show, BPC matched all previous best-known solutions and unveiled 11 new best-known solutions in all the replicates. On the subset of instances with 20 customers, both MSH and SLNS matched all the best-known solutions. However, as the number of customers increases, the quality of the solutions found decreases.

Table 2.2 compares the performance of BPC in the best performing replicate against the benchmark algorithms in terms of the optimality gap. To compute the optimality gap we used the lower bound found by our BPC. Similar to Table 2.1 the results are grouped by instance size. Columns 2, 5, 8, and 11 show the number of optimal solutions found by each algorithm. Columns 3, 6, 9, and 12 contain the average optimality gap. Columns 4, 7, 10, and 13 show the maximum optimality gap.

Table 2.2: Assessing optimality on the Coindreau et al. (2019) instances.

$\|C\|$	VNS			MSH			SLNS			BPC		
	#Opt.	Avg. Δ	Max. Δ	#Opt.	Avg. Δ	Max. Δ	#Opt.	Avg. Δ	Max. Δ	#Opt.	Avg. Δ	Max. Δ
20	1/10	4.09%	8.87%	10/10	0.00%	0.00%	10/10	0.00%	0.00%	10/10	0.00%	0.00%
30	0/10	3.13%	5.89%	7/10	0.01%	0.04%	8/10	0.00%	0.04%	10/10	0.00%	0.00%
40	3/10	1.23%	5.31%	6/10	0.15%	0.69%	7/10	0.22%	1.39%	10/10	0.00%	0.00%
50	2/10	2.22%	6.11%	2/10	1.00%	2.60%	4/10	0.23%	0.93%	9/10	0.09%	0.91%
Total/Avg.	6/40	2.67%	6.54%	25/40	0.29%	0.83%	29/40	0.11%	0.59%	39/40	0.02%	0.23%

Note that BPC is the first to prove optimality for 39 (out of 40) instances. Moreover, the average optimality gap of the solutions found by BPC is 0.02%. Additionally, the maximum optimality gap on the unsolved instances is 0.91%. With regard to the metaheuristics, SLNS has the best performance finding solutions with an average optimality gap of 0.11%.

Finally, Table 2.3 compares the performance of each algorithm in terms of computational efficiency. Each row corresponds to an instance size. Columns 2, 3, 4, and 5 show the average runtime in seconds reported by each algorithm. Columns 6 and 7 show the minimum and maximum CPU time employed by the BPC.

Table 2.3: Computational times on the Coindreau et al. (2019) instances.

$\|C\|$	VNS	MSH	SLNS	BPC		
	Avg. CPU (s)	Avg. CPU (s)	Avg. CPU (s)	Avg. CPU (s)	Min. CPU (s)	Max. CPU (s)
20	71.0	13.4	15.0	28.1	22.1	40.1
30	381.0	35.9	30.0	75.9	51.3	127.7
40	1993.0	56.1	60.0	288.7	100.2	1340.8
50	6779.0	110.5	120.0	1877.4	276.6	7200.0
Average	2306.0	54.0	56.3	567.5	112.5	2177.1

While a perfect head-to-head comparison is hard to make because of differences in the programming languages and testing environment, the results suggest that in the subset of instances with 20 and 30 customers, BPC is close to match the performance of the state-of-the-art matheuristics. On average BPC uses less than 10 minutes.

2.5.3 Assessing the impact of the initialization step and the pruning strategies

As our proposed method uses a metaheuristic to initialize the set of columns, it is only logical to assess the performance of our BPC without this component. With this in mind, we ran our BPC while only using one iteration of MSH. This version of the algorithm is labeled as BPC-W. Moreover, to measure the impact of the problem-specific algorithmic components that we designed, we ran our BPC with a version of the PA that does not include the path completion and the subtour fixing strategies. This version of our BPC is labeled as BPC-O.

Table 2.4 compares the performance of the branch-price-and-cut algorithms described above in terms of solution quality. Each row corresponds to a combination between a version of our BPC algorithm and an instance size. Recall that each algorithm was tested on every instance across five replicates in which the initial solution is modified. Column 3 shows the average number of best-known solutions found by each algorithm. Column 4 shows the average number of optimal solutions found by each algorithm. Columns 5, 6 and 7 present the average, minimum, and maximum

computational time in seconds used by each algorithm. Finally, columns 8, 9 and 10 show the average, minimum, and maximum number of columns.

Table 2.4: Solution quality of the BPC variants with $\zeta = 5$ km.

Algorithm	$\|C\|$	Avg. #BKS	Avg. # Optimal	CPU time (s)			# Columns		
				Avg.	Min	Max	Avg.	Min	Max
BPC	20	10/10	10/10	28.11	22.12	40.13	219.30	85	461
	30	10/10	10/10	75.90	51.29	127.67	393.74	92	693
	40	10/10	10/10	288.72	100.17	1340.78	1647.88	387	6050
	50	10/10	8.8/10	1877.42	276.60	7200.00	1864.00	779	5426
BPC-O	20	10/10	10/10	27.19	21.32	39.61	221.54	85	461
	30	10/10	10/10	74.07	50.60	138.73	392.72	92	664
	40	10/10	10/10	317.17	96.14	1755.69	1562.46	486	6824
	50	9/10	8.2/10	2026.22	279.66	7200.00	1807.70	646	3934
BPC-W	20	10/10	10/10	14.43	4.15	41.62	2246.14	555	5155
	30	10/10	10/10	118.84	28.60	342.65	12998.62	1586	35468
	40	10/10	10/10	1018.09	242.95	4219.65	5871.14	1874	39067
	50	8.8/10	8.4/10	2346.02	354.07	7200.00	10070.38	2312	53989

Note that BPC is the only algorithm that finds the best-known solution for every instance in all the replicates. Moreover, it is the algorithm that on average provides the highest number of optimal solutions. Nevertheless, note that BPC-W is capable of finding high quality solutions and of proving optimality of at least 38 out of 40 instances. This shows that the BPC solution quality is not exclusively due to the initialization procedure. In addition, note that BPC-O is not capable of finding one of the best-known solutions before reaching the running time limit, thus showing the importance of including problem-specific pruning strategies inside the PA.

With respect to the computational times, BPC is on average faster than its counterparts, especially on the subset of instances with 40 and 50 customers, where both the initialization procedure and the additional pruning strategies improve the algorithm's performance. However, in the subset of instances with 20 customers, it seems that the overhead incurred by the initialization procedure does not pay off, as BPC-W is faster than BPC and delivers solutions of equivalent quality.

2.5.4 A new data set

In the previous experiments, we showed that our algorithm is able to optimally solve most of the 50-node instances in the PLRP set. It is worth recalling that these instances are the largest benchmarks available for our problem. To test the limits of our method, we designed a new set of larger instances. To build them, we followed the procedure described by Coindreau et al. (2019). Each instance considers a number of customers n in the set $\{60, 70, 80, 90\}$. For each value of n, we randomly generate 10 instances in which the (x,y)-coordinates of the customers and the depot are taken from a continuous uniform distribution on the intervals $[0, 10] \times [0, 10]$. In this new set of instances, the number of available workers k is set to the minimum number of workers needed to serve all the customers. This value is calculated using the left-hand side of constraint 2.13. All the other parameters described in Section 2.5.1 remain the same. The new set is publicly available at `www.vrp-rep.org` (VRP-REP-ID 2023-0001).

Table 2.5 describes the performance of the BPC algorithm on the new set of larger instances. Each row corresponds to an instance size. Column 2 shows the number of optimal solutions found by the algorithm. Columns 3, 4 and 5 report the average, minimum, and maximum optimality gap, respectively. Columns 6, 7 and 8 contain the average, minimum, and maximum CPU time employed by the BPC. Columns 9, 10 and 11 report the average, minimum, and maximum gap in the objective function of the solutions found by BPC with respect to those delivered by the initialization method, MSH. More specifically, for each instance in the set, the gap was computed as

$$\frac{f(BPC) - f(MSH)}{f(MSH)}, \tag{2.15}$$

where $f(\cdot)$ denotes the objective function of the solution delivered by a given approach. For these experiments, the time limit is set to two hours. All the parameters used by MSH (e.g., $T = 2500$, $Q = 60$) were fixed to their standard values (see Section 2.4.3).

Table 2.5: Branch-price-and-cut performance on larger PLRP instances.

$\|C\|$	# Opt.	Optimality gap			CPU time (s)			Δ vs initial solution		
		Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
60	3	1.62%	0.00%	3.66%	5460.14	396.02	7200.00	-2.04%	-5.83%	0.00%
70	0	3.43%	2.17%	5.39%	7200.00	7200.00	7200.00	-0.17%	-0.58%	0.00%
80	0	5.66%	4.07%	8.21%	7200.00	7200.00	7200.00	-1.21%	-5.17%	0.00%
90	0	5.93%	3.73%	8.26%	7200.00	7200.00	7200.00	-0.49%	-2.20%	0.00%
Total/Avg.	3	4.16%	2.49%	6.38%	6765.03	5499.01	7200.00	-0.98%	-3.44%	0.00%

As expected, the performance of the BPC decreases when the number of customers $|C|$ increases. Nevertheless, BPC is capable of proving optimality for 3 out of 10 instances with 60 customers within the time limit. Moreover, the average optimality gap in the subset of instances with 60 customers is 1.62%. Regarding the subset of instances with 70, 80, and 90 customers, BPC is not capable of closing the optimality gap for any of the instances within the time limit. While performing our experiments, we noticed that most of the time spent by BPC is used to try to improve the lower bound. Indeed, in this new set of instances, the average improvement of the solutions found by MSH is only 0.98%. Thus one possible direction to extend the reach of our method to 70-plus-customer instances would be to devise new valid inequalities to strengthen the lower bound faster. For completeness, Table B.2 in Appendix B shows the objective function value of the solutions found by MSH and BPC on each individual instance. It also reports the lower bound found by BPC before reaching the time limit.

2.5.5 Analyzing the pricing problem algorithm

Most modern BPC algorithms for vehicle routing solve the pricing problem using labeling algorithms that rely on the NG-path relaxation proposed by (Baldacci et al. 2011). This relaxation consists in defining a neighborhood $\mathcal{N}_i$ for every customer $i \in C$ that includes the X closest customers to i and the customer itself. An NG-path can include cycles starting and ending at customer j if and only if there exists a customer i in the cycle such that $j \in \mathcal{N}_i$. Thus, such a cycle is forbidden if and only if $j \in \mathcal{N}_i$ for every customer i it contains. The higher the value of X, the tighter

the relaxation will be. Due to its success in column generation based algorithms, we believe it is interesting to compare the performance of our adaptation of the PA against a classical single-directional labeling algorithm implementing the NG-path relaxation. The implemented labeling algorithm uses dominance rules to discard labels. It also exploits an adapted version of the infeasibility, bounds, and rollback pruning strategies. Moreover, during the first iterations, the labeling algorithm is applied heuristically following the same logic described in Section 2.4.2. In addition, at every iteration, we first relax the dominance rules by ignoring conditions (33)-(34). An in-depth description of the algorithm is available in Appendix A.

To compare the performance of the PA and the labeling algorithm, we solve the RSCP without applying any cuts or branching decisions. In the remainder of this section, the version of the algorithm using the PA to solve the pricing problem is labeled CG-PA, and that using the labeling algorithm is labeled CG-NG-X. For these experiments, we solve the instances with 20 and 30 customers under two settings. First, the case in which walking is not allowed, namely $\zeta = 0$ km. Second, the case in which walking is allowed, namely $\zeta = 5$ km. We consider these two settings to isolate the impact of including park-and-loop routes on both pricing algorithms.

Table 2.6 compares the performance of each column generation algorithm while solving the RSCP when walking is not allowed. Each row in the table corresponds to a pricing algorithm and a number of customers. Columns 3, 4 and 5 show the average, minimum, and maximum lower bound produced by each approach. Columns 6, 7 and 8 present the average, minimum, and maximum computational time in seconds. Finally, columns 9, 10 and 11 show the average, minimum, and maximum number of columns found by the corresponding pricing algorithm.

As the results show, CG-PA provides on average a better (higher) lower bound than the CG algorithms that use the NG-path relaxation. However, in this subset of instances the CG algorithms using the NG-path relaxation solve the RSCP faster. With respect to the number of columns, CG-PA requires on average a lower number of columns. These results are expected, as both CG-NG-10 and CG-NG-7 are solving

Table 2.6: Performance of the CG algorithms solving the RSCP root node with $\zeta = 0$.

CG algorithm	$\|C\|$	Lower bound			CPU time (s)			# Columns		
		Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
CG-PA	20	41.85	33.64	46.01	21.53	17.61	24.66	321.00	117	827
	30	**52.35**	**47.91**	55.84	43.49	37.20	49.84	538.06	167	1046
CG-NG-10	20	41.84	33.64	46.01	11.38	10.37	13.30	548.30	125	2592
	30	**52.22**	**47.78**	55.84	21.72	18.58	26.35	769.04	215	1762
CG-NG-7	20	41.75	33.43	46.01	11.53	10.34	15.38	758.88	171	3565
	30	**52.11**	**47.49**	55.54	21.90	18.88	26.08	938.98	209	2364

a relaxation of the pricing problem. Moreover, without the presence of walking subtours, the subtour fixing strategy used by the PA does not have an impact on the algorithm's performance.

Table 2.7 compares the performance of each column generation algorithm while solving the RSCP when walking is allowed. Similarly, each row in the table corresponds to a pricing algorithm and a number of customers. Columns 3, 4 and 5 show the average, minimum, and maximum lower bound delivered by each approach. Columns 6, 7 and 8 present the average, minimum, and maximum computational time in seconds. Finally, columns 9, 10 and 11 show the average, minimum, and maximum number of columns found by the corresponding pricing algorithm.

Table 2.7: Performance of the pricing algorithms solving the root node with $\zeta = 5$ km.

CG algorithm	$\|C\|$	Lower bound			CPU time (s)			# Columns		
		Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max
CG-PA	20	36.64	29.61	42.28	25.79	21.10	30.95	194.06	90	503
	30	**45.16**	**40.64**	49.59	**61.02**	**50.78**	**78.54**	349.54	84	672
CG-NG-10	20	36.62	29.61	42.28	35.94	21.36	69.54	287.70	95	666
	30	**45.08**	**40.21**	49.59	**370.70**	**127.06**	**918.51**	475.64	72	1036
CG-NG-7	20	36.51	28.90	42.28	32.81	19.23	63.07	349.28	89	641
	30	**44.89**	**40.00**	49.59	**273.68**	**94.67**	**749.04**	591.86	78	1195

Note that the CG algorithms using the NG-paths relaxation provide lower bounds of lower quality in comparison with the bounds found by CG-PA. This can have a large impact on the solution algorithm as branch-and-bound nodes can be pruned early. Moreover, the computational time required to solve the RSCP is signifi-

cantly larger for the NG-paths based algorithms. Indeed, considering the subset of instances with 30 customers, on average CG-PA takes 61.02 seconds to solve the RSCP while CG-NG-10 and CG-NG-7 take 370.70 and 270.68 seconds, respectively. Although we can only speculate, we suspect there are two main reasons for these differences in performance. First, recall that the PA executes a depth-first search, while the labeling algorithm performs a breadth-first search. Thus, the PA tends to find promising routes faster and, as such, it significantly benefits from the increased strength of the bounds pruning strategy. Secondly, as opposed to the PA, the labeling algorithm relies heavily on dominance rules in order to discard labels. However, when walking subtours are allowed, labels are only comparable if they have the same parking status (i.e., same parking spot), which is often not the case, especially when the number of customers is high. As a result, in our experiments we observed that the size of the labels queue considerably grows in comparison with the case in which walking is not allowed.

To push the analysis forward, we implemented a BPC algorithm that uses the labeling algorithm to solve the pricing problem and reuses all the branching and cutting machinery in our BPC algorithm. For quick reference, we denote this alternative method as BPC-NG-X. We ran experiments on the Coindreau et al. (2019) instances setting X equal to 7. All other parameters for both two algorithms (e.g., $\varphi = 5$, $\varepsilon = 0.1$, $\Upsilon = 10$) were fixed to their standard values (see Section 2.5.1). Table 2.8 describes the performance of both BPC algorithms. Each row corresponds to an instance size. Columns 2 and 4 show the number of instances in which the corresponding BPC algorithm was able to prove optimality. Columns 3 and 5 show the number of best-known solutions found by each BPC algorithm. Columns 6, 7 and 8 contain the average, minimum, and maximum computational time used by BPC-NG-7. The computational times of BPC are shown in Table 2.4.

As the results show, our BPC outperforms BPC-NG-7 in both solution quality and running times. More specifically, BPC-NG-7 only proves optimality for 19/40 instances while BPC proves optimality for 39/40 instances. Moreover, BPC-NG-7

Table 2.8: Performance of BPC algorithms on the Coindreau et al. (2019) instances.

$\|\mathcal{C}\|$	BPC		BPC-NG-7				
	# Opt.	# BKS	# Opt.	# BKS	Avg. CPU (s)	Min. CPU (s)	Max. CPU (s)
20	10/10	10/10	10/10	10/10	182.52	25.92	1218.00
30	10/10	10/10	7/10	9/10	2795.20	129.32	7200.00
40	10/10	10/10	2/10	6/10	6572.29	2640.12	7200.00
50	9/10	10/10	0/10	0/10	7200.00	7200.00	7200.00
Total/Avg.	39/40	40/40	19/40	25/40	4187.50	2498.84	5704.50

is only capable of matching 25/40 of the best-known solutions within the time limit. Regarding the computational times, BPC-NG-7 uses on average 4187.50 seconds to solve PLRP instances while BPC only uses 567.5. Thus, the performance drop observed by changing the algorithm that solves the pricing problem is significant. Furthermore, it seems that BPC-NG-7 particularly struggles on the subset of instances with 50 customers, in which even matching the best-known solution can be challenging.

2.5.6 Assessing the importance of introducing park-and-loop routes

Introducing park-and-loop routes helps to decrease the driven distance. To assess their potential impact, we compare the objective function of the best-known solutions while varying the maximum walking distance. Namely, we consider the vehicle routing configuration (i.e., $\zeta = 0$) and the park-and-loop configuration with ζ equal to 5 or 10 kilometers.

Table 2.9 compares the average driven distance of each configuration. Each row corresponds to an instance size. Column 2 shows the average driven distance without allowing walking subtours. Columns 3 and 5 show the average driven distance while imposing a limit of 5 or 10 kilometers on the total walking distance respectively. Finally, Columns 4 and 6 show the average savings gained by introducing walking subtours.

As the results show, introducing walking subtours decreases the driven distance

Table 2.9: Average driven distance while varying the maximum walking distance.

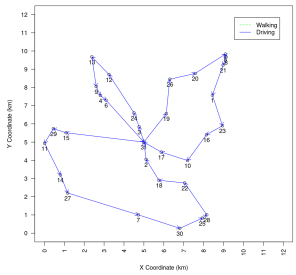
# Customers	VRP	PLRP ($\zeta = 5$)		PLRP ($\zeta = 10$)	
	Avg. km	Avg. km	% Δ vs VRP	Avg. km	% Δ vs VRP
20	42.08	36.71	-12.8%	32.63	-22.5%
30	53.35	45.32	-15.0%	38.04	-28.7%
40	60.82	56.84	-6.5%	54.97	-9.6%
50	69.38	61.80	-10.9%	60.43	-12.9%
Average	56.41	50.17	**-11.3%**	46.52	**-18.4%**

by 11.3% and 18.4% on average. As expected, increasing the walking distance limit allows for larger walking subtours and, in turn, for greater savings. Figure 2.3 shows the best-known solutions for instance *30_A_1*. When walking subtours are not allowed, the total driven distance is 48.81 km. If walking subtours are allowed, the driven distances are 40.73 km and 32.93 km while setting ζ to 5 and 10 kilometers, respectively. As stated before, major savings can be achieved by introducing park-and-loop routes. As this figure shows, increasing the maximum walking distance has an impact on the size and length of the walking subtours present in each route.

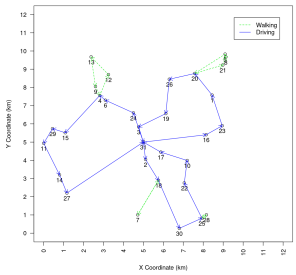
We have developed a website available at `https://chairelogistique.hec.ca/en/scientific-data/` where researchers can download the instances, access instance-by-instance results, and upload their own solutions in order to encourage future research on this problem and make comparisons with our results easier.

2.6 Concluding remarks

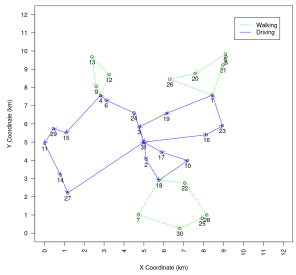
In this paper, we presented a branch-price-and-cut algorithm for solving the PLRP. To do so, we formulated the PLRP as a set covering problem that considers a large set of paths. To solve this model, we use a column generation approach that unveils promising paths that favor the objective of minimizing the total cost. The master problem selects paths to serve all customers, while the pricing problem generates feasible paths. We leveraged a tailored and improved version of the pulse algorithm to solve an elementary resource constrained shortest path with park-and-loop, that



(a) VRP configuration



(b) PLRP with $\zeta = 5$ km



(c) PLRP with $\zeta = 10$ km

Figure 2.3: Solutions for instance *30_A_1*.

includes additional pruning strategies and allows for considering the subset row inequalities. Using this new version of the pulse algorithm improves the performance of the branch-price-and-cut algorithm.

We compared our branch-price-and-cut algorithm with the state-of-the-art algorithms for solving a related problem called the vehicle routing problem with transportable resources without carpooling. The proposed algorithm was capable of finding all the previously best-known solutions. Moreover, it found 11 previously unknown optimal solutions. In addition, our method is the first capable of solving 39 instances to optimality out of the 40 instances that composed the testbed. Our experiments also show the advantages of providing a high quality pool of columns as a warm start as it significantly decreases the computational effort required by the algorithm.

We also showed the benefits of introducing a park-and-loop structure in the

routing plan, as it allows decreasing the total driven distance. To encourage future research on the PLRP, we developed an online tool that allows the members of the community to visualize the best-known solutions and upload their own solutions for checking and plotting. Future research should focus on extending the branch-price-and-cut algorithm to solve the PLRP with time windows. Moreover, improving the performance of the pricing problem by implementing bidirectional search strategies and other problem-specific pruning strategies seems promising to decrease the computational time needed by each CG iteration.

References

Arslan, O., O. Jabali, and G. Laporte (2018). "Exact solution of the evasive flow capturing problem". *Operations Research* 66.6, pp. 1625–1640.

Baldacci, R., A. Mingozzi, and R. Roberti (2011). "New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem". *Operations Research* 59.5, pp. 1269–1283.

Belenguer, J., E. Benavent, A. Martinez, C. Prins, C. Prodhon, and J. Villegas (2016). "A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots". *Transportation Science* 50, pp. 735–749.

Bolívar, M. A., L. Lozano, and A. L. Medaglia (2014). "Acceleration strategies for the weight constrained shortest path problem with replenishment". *Optimization Letters* 8.8, pp. 2155–2172.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2022). "The doubly open park-and-loop routing problem". *Computers & Operations Research* 143, p. 105761.

Cabrera, N., A. L. Medaglia, L. Lozano, and D. Duque (2020). "An exact bidirectional pulse algorithm for the constrained shortest path". *Networks* 76.2, pp. 128–146.

Chao, I.-M. (2002). "A tabu search method for the truck and trailer routing problem". *Computers and Operations Research* 29, pp. 33–51.

Coindreau, M.-A., O. Gallay, and N. Zufferey (2019). "Vehicle routing with transportable resources: Using carpooling and walking for on-site services". *European Journal of Operational Research* 279, pp. 996–1010.

Contardo, C. and R. Martinelli (2014). "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". *Discrete Optimization* 12, pp. 129–146.

Corredor-Montenegro, D., N. Cabrera, R. Akhavan-Tabatabaei, and A. L. Medaglia (2021). "On the shortest α-reliable path problem". *Top* 29.1, pp. 287–318.

Costa, L., C. Contardo, and G. Desaulniers (2019a). "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing". *Transportation Science* 53, pp. 946–985.

Costa, L., C. Contardo, and G. Desaulniers (2019b). "Exact branch-price-and-cut algorithms for vehicle routing". *Transportation Science* 53.4, pp. 946–985.

Derigs, U., M. Pullmann, and U. Vogel (2013). "Truck and trailer routing - Problems, heuristics and computational experience". *Computers and Operations Research* 40, pp. 536–546.

Desrosiers, J., M. Lübbecke, G. Desaulniers, and J. B. Gauthier (2024). *Branch-and-Price*. Les Cahiers du GERAD G-2024-36. GERAD, Montréal QC H3T 2A7, Canada: Groupe d'études et de recherche en analyse des décisions, pp. 1–657.

Duque, D., L. Lozano, and A. L. Medaglia (2014). "Solving the orienteering problem with time windows via the pulse framework". *Computers and Operations Research* 54, pp. 168–176.

Duque, D., L. Lozano, and A. L. Medaglia (2015). "An exact method for the biobjective shortest path problem for large-scale road networks". *European Journal of Operational Research* 242.3, pp. 788–797.

Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen (2004). "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". *Networks* 44.3, pp. 216–229.

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

Le Colleter, T., D. Dumez, F. Lehuédé, and O. Péton (2023). "Small and large neighborhood search for the park-and-loop routing problem with parking selection". *European Journal of Operational Research* 308.3, pp. 1233–1248.

Li, H., H. Wang, J. Chen, and M. Bai (2020). "Two-echelon vehicle routing problem with time windows and mobile satellites". *Transportation Research Part B: Methodological* 138, pp. 179–201.

Lin, S., V. Yu, and S. Chou (2009). "Solving the truck and trailer routing problem based on a simulated annealing heuristic". *Computers and Operations Research* 36, pp. 1683–1692.

Lozano, L., D. Duque, and A. L. Medaglia (2016). "An exact algorithm for the elementary shortest path problem with resource constraints". *Transportation Science* 50.1, pp. 348–357.

Lozano, L. and J. C. Smith (2017). "A backward sampling framework for interdiction problems with fortification". *INFORMS Journal on Computing* 29.1, pp. 123–139.

Lübbecke, M. E. and J. Desrosiers (2005). "Selected topics in column generation". *Operations Research* 53.6, pp. 1007–1023.

Marques, G., R. Sadykov, J.-C. Deschamps, and R. Dupas (2020). "An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem". *Computers & Operations Research* 114, p. 104833.

Martinez-Sykora, A., F. McLeod, C. Lamas-Fernandez, T. Bektaş, T. Cherrett, and J. Allen (2020). "Optimised solutions to the last-mile delivery problem in London using a combination of walking and driving". *Annals of Operations Research* 295.2, pp. 645–693.

Montoya, A., C. Guéret, J. E. Mendoza, and J. G. Villegas (2016). "A multi-space sampling heuristic for the green vehicle routing problem". *Transportation Research Part C: Emerging Technologies* 70, pp. 113–128.

Parragh, S. N. and J.-F. Cordeau (2017). "Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows". *Computers and Operations Research* 83, pp. 28–44.

Pessoa, A., R. Sadykov, E. Uchoa, and F. Vanderbeck (2013). "In-Out separation and column generation stabilization by dual price smoothing". *Experimental Algorithms*. Ed. by V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 354–365. ISBN: 978-3-642-38527-8.

Reed, S., A. M. Campbell, and B. W. Thomas (2024). "Does parking matter? The impact of parking time on last-mile delivery optimization". *Transportation Research Part E: Logistics and Transportation Review* 181, p. 103391.

Restrepo, M. I., L. Lozano, and A. L. Medaglia (2012). "Constrained network-based column generation for the multi-activity shift scheduling problem". *International Journal of Production Economics* 140.1, pp. 466–472.

Rothenbächer, A.-K., M. Drexl, and S. Irnich (2018). "Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows". *Transportation Science* 52.5, pp. 1174–1190.

Schrotenboer, A. H., E. Ursavas, and I. F. Vis (2019). "A branch-and-price-and-cut algorithm for resource-constrained pickup and delivery problems". *Transportation Science* 53.4, pp. 1001–1022.

Semet, F. (1995). "A two-phase algorithm for the partial accessibility constrained vehicle routing problem". *Annals of Operations Research* 61, pp. 45–65.

Sheuerer, S. (2006). "A tabu search heuristic for the truck and trailer routing problem". *Computers and Operations Research* 33, pp. 894–909.

Tamke, F. and U. Buscher (2021). "A branch-and-cut algorithm for the vehicle routing problem with drones". *Transportation Research Part B: Methodological* 144, pp. 174–203.

Uchoa, E., A. Pessoa, and L. Moreno (2024). *Optimizing with Column Generation: Advanced Branch-Cut-and-Price Algorithms (Part I)*. Tech. rep. L-2024-3. Cadernos do LOGIS-UFF, Universidade Federal Fluminense, Engenharia de Produção.

Villegas, J., C. Prins, A. Medaglia, and N. Velasco (2010). "GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots". *Engineering Applications of Artificial Intelligence* 23, pp. 780–794.

Villegas, J., C. Prins, A. Medaglia, and N. Velasco (2013). "A matheuristic for the truck and trailer routing problem". *European Journal of Operational Research* 230, pp. 231–244.

Villegas, J., C. Prins, C. Prodhon, A. Medaglia, and N. Velasco (2011). "A GRASP with evolutionary path relinking for the truck and trailer routing problem". *Computers and Operations Research* 38, pp. 1319–1334.

Yamín, D., A. L. Medaglia, and A. A. Prakash (2022). "Exact bidirectional algorithm for the least expected travel-time path problem on stochastic and time-dependent networks". *Computers & Operations Research* 141, p. 105671.

Yin, Y., D. Li, D. Wang, J. Ignatius, T. Cheng, and S. Wang (2023). "A branch-and-price-and-cut algorithm for the truck-based drone delivery routing problem with time windows". *European Journal of Operational Research* 309.3, pp. 1125–1144.

Zhou, H., H. Qin, C. Cheng, and L.-M. Rousseau (2023). "An exact algorithm for the two-echelon vehicle routing problem with drones". *Transportation Research Part B: Methodological* 168, pp. 124–150.

Chapter 3

The Workforce Scheduling and Routing Problem with Park-and-loop

Abstract

This paper introduces formulations and an exact algorithm for the workforce scheduling and routing problem with park-and-loop. This problem extends the standard workforce scheduling and routing problem by allowing the use of walking subtours in the routes. We introduce a compact arc-based formulation as well as a path-based formulation with an exponential number of variables. To efficiently solve the latter, we propose a branch-price-and-cut algorithm that leverages state-of-the-art techniques, including a tailored version of the pulse algorithm to solve the pricing problem and the separation of subset row inequalities to strengthen the lower bound. We report on computational experiments carried out on a set of instances with up to 75 tasks adapted from the literature. The results show that our method systematically outperforms a standard MIP solver, proving optimality for 241 out of 324 instances. We also report experiments on the closely-related service technician routing and scheduling problem, where our method delivered 12 new best solutions on a 54-instance testbed from the literature.

3.1 Introduction

In this paper, we study the workforce scheduling and routing problem with park-and-loop (WSRP-PL). While our initial inspiration came from a real-world application in France, similar problems are encountered by utility and service companies around the world. In our problem, a company must perform a set of on-site tasks (e.g., connection to utility grids, troubleshooting, meter reading). Each task has an associated duration and an associated time window. In addition, depending on its nature, a task may require one or more skills, each at a potentially different level of proficiency. Tasks are executed either by the company's workforce or by a third party. Every worker in the force masters a subset of skills, each at a given proficiency level. Workers can work individually or in teams. To simplify the narrative, we assume that workers always work in teams (but a team can comprise just one worker). Teams depart from and must return to a single depot within working hours driving a vehicle (i.e., a car). Because many customers are located in densely populated areas, in which access to parking may be limited, workers can walk between nearby locations (after safely parking the car). The WSRP-PL consists in building a plan to execute all the tasks while minimizing the total operational cost. The latter is composed of (i) the outsourcing cost of tasks assigned to the third party and (ii) the cost of the total distance driven by the internal teams. A plan is defined by the assignment of workers to teams for the day and the routing of the vehicles driven by the teams.

Our problem is closely related to the park-and-loop routing problem (PLRP) and the workforce scheduling and routing problem (WSRP), both of which are NP-Hard. The PLRP is a variation of the vehicle routing problem (VRP), where routes consist of a primary tour completed using a vehicle, along with sub-tours performed on foot after parking the vehicle. In the PLRP, and most of its variants, the route duration and total walking distance are bounded. Coindreau et al. (2019) introduced the vehicle routing problem with transportable resources (VRPTR). In this variant of

the PLRP, workers are allowed to share a vehicle (i.e., carpool). To address this problem, the authors proposed a variable neighborhood search (VNS) algorithm. They assessed their algorithm using a set of new instances containing up to 50 customers. The experiments showed that their VNS could provide solutions for all instances within a computing time of 10 hours. It is worth noting that, in contrast to our WSRP-PL, in the VRPTR, workers are considered identical and capable of fulfilling all tasks.

Cabrera et al. (2022) introduced the doubly open park-and-loop routing problem (DOPLRP). In this variant of the PLRP workers initiate and conclude their routes at customer locations. To address this problem, they proposed a customized implementation of the multi-space sampling heuristic (Mendoza and Villegas 2013). They conducted experiments using a dataset of real-world instances provided by a French utility, featuring up to 3,000 customers. Their findings demonstrated that their method consistently produced solutions that resulted in significant cost savings compared to those generated by the company's routing software. Additionally, they applied their method to instances introduced by Coindreau et al. (2019) and were able to enhance 32 out of the 40 previously best-known solutions. There are two key distinctions between our problem and their DOPLRP. First, in DOPLRP, tasks do not have time windows or skill requirements. Consequently, workers are not required to form teams. Second, in our WSRP-PL, a fundamental requirement is that all routes must both start and end at the depot.

In a similar vein, Le Colleter et al. (2023) conducted a study on the park-and-loop routing problem with parking selection (PLRP-PS). In this particular variant of the PLRP, the vehicle can only be parked at predefined parking locations. The authors introduced a small and large neighborhood search metaheuristic. To enhance the algorithm's efficiency, they implemented operators specifically designed for selecting parking spots. The algorithm unveiled eight new best-known solutions for the Coindreau et al. (2019) instances. These solutions were later improved or confirmed as optimal by Cabrera et al. (2023), who introduced a branch-price-and-

cut (BPC) algorithm for the PLRP. Through their algorithm, they found optimal solutions for 39 out of the 40 Coindreau et al. (2019) instances. The key component of their method is the pulse algorithm (PA), employed for solving the pricing problem. They improved the classical PA with problem-specific pruning strategies that significantly accelerated the algorithm. It is worth noting that their BPC is not equipped to handle task skill requirements, team formation, or time windows.

The workforce scheduling and routing problem (WSRP) combines elements from both scheduling and routing problems. In the scheduling component of this problem, the objective is to assign workers (e.g., technicians, nurses, and security guards) to provide a service or complete tasks for customers. When making these worker assignments, various features are taken into account, including skills compatibility (Braekers et al. 2016; Chen et al. 2016; Kovacs et al. 2012), team formation (Bredström and Rönnqvist 2008; Zamorano et al. 2018), multiple time periods (Guastaroba et al. 2021; Tricoire et al. 2013), and precedence constraints (Goel and Meisel 2013; Pereira et al. 2020), among others. On the other hand, the routing component involves designing a set of routes that workers use to travel between different locations. This routing process takes into consideration factors such as customer time windows, the presence of multiple depots, and the potential use of alternative transportation modes. For further exploration of applications and variants of the WSRP, interested readers are referred to Castillo-Salazar et al. (2016) and Paraskevopoulos et al. (2017) for a comprehensive review.

Our problem is a generalization of the technician routing and scheduling problem (STRSP) introduced by Kovacs et al. (2012), one of the most studied WSRP variants. They solved two versions of the problem. The *no-team* version in which routes are carried out by individual workers, and the *team* version in which routes are designed for multiple workers. They proposed an adaptive large neighborhood search (ALNS) heuristic that includes a set of classical destroy and repair operators. Using this methodology, the authors provided high-quality solutions to instances with up to 100 tasks in less than two minutes. The authors also proposed a mathe-

matical formulation that they solve by means of a mixed integer problem solver (i.e., CPLEX). They tested their exact method on instances with 25 tasks. As opposed to the WSRP-PL, in the STRSP workers are only allowed to drive between locations.

Most of the work on the no-team version of the STRSP has focused on heuristic algorithms (Gu et al. 2022; Xie et al. 2017; Zhou et al. 2020). Xie et al. (2017) proposed an iterated local search (ILS) algorithm that uses cleverly designed neighborhood structures. Their method was evaluated against the ALNS of Kovacs et al. 2012, showing an improved performance in both solution quality and speed. Building on the algorithm by Xie et al. (2017), Zhou et al. 2020 presented an iterated local search with hybrid neighborhood search (ILS-HNS) algorithm. This algorithm switches between small and large neighborhoods, which allows the algorithm to escape local optima. The proposed algorithm improved 12 of the best known solutions. Similarly, Gu et al. (2022) presented a Lagrangian iterated local search (L-ILS) algorithm which significantly outperforms ILS-HNS. Their study sets L-ILS as the state-of-the-art algorithm. Note, however, that neither of these methods is exact. Moreover, neither of these methods is capable of solving the team version of the STRSP.

Our study contributes to the existing literature in several key ways. First, we introduce the workforce scheduling and routing problem with park-and-loop, a problem that arises at the intersection of two challenging combinatorial problems: the PLRP and the WSRP. Both of these problems have gained increasing practical relevance due to concerns such as labor shortages and carbon emissions. Second, to tackle this challenging problem, we have developed an exact Branch-Price-and-Cut (BPC) algorithm that leverages state-of-the-art techniques. Our computational experiments demonstrate the superior efficiency of our BPC algorithm when compared to a standard mixed-integer programming (MIP) solver. Third, we have applied our algorithm to provide optimality certificates for 24 instances on a standard testbed for the no-team version of the STRSP. Out of these, 12 represent new best-known solutions. Finally, we have created an online tool that enables researchers to visual-

ize and download all the solutions and instances reported in our paper, enhancing accessibility and usability for the research community.

This paper is organized as follows. Section 3.2 formally introduces the WSRP-PL. Section 3.3 presents a path-based formulation for the problem. Section 3.4 describes the proposed branch-price-and-cut algorithm. Section 3.5 presents the computational experiments. Finally, Section 3.6 presents the conclusions and outlines potential paths for future research.

3.2 Problem description and arc-based formulation

The WSRP-PL can be formally defined on a complete and directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of directed arcs. The set of nodes comprises a start depot $\overline{0}$, an end depot $\underline{0}$, and the set of tasks $\mathcal{C} = \{1, ..., n\}$. Note that, $\overline{0}$ and $\underline{0}$ can represent the same or distinct geographical locations. Arcs in $\mathcal{A}$ represent the connections between two tasks or between a task and the depot. To perform the tasks, a set of workers $\mathcal{W} = \{1, ..., m\}$ are assigned to teams in the set $\mathcal{T}$. Note that $|\mathcal{T}|$ must be an upper bound on the maximum number of teams to form. A maximum of ω workers can be assigned to a team. Each team $t \in \mathcal{T}$ departs from and arrives to the depot after performing its route. Each route has a maximum duration ϕ. Teams can drive or walk between locations. Accordingly, each arc $(i, j) \in \mathcal{A}$ has four main attributes: the driving distance μ_{ij}, the driving time τ_{ij}, the walking distance δ_{ij}, and the walking time η_{ij}. The maximum distance that can be traveled on foot between two points is θ. Moreover, the maximum distance that can be traveled by a team on foot in one day is ζ. Driving the car involves a variable cost c^v per unit of distance while walking is assumed to be free of charge.

Each task $i \in \mathcal{C}$ has a duration s_i and an associated time window indicating possible visit times. Let $[a_i, b_i]$ be the earliest and latest starting time of task $i \in \mathcal{C}$. Also, let f_i be the outsourcing cost of task $i \in \mathcal{C}$. Skill requirements are represented by ν_{iql}, an integer parameter stating the number of workers with the skill $q \in \mathcal{Q}$ with

at least a proficiency level $l \in \mathcal{L}$ that the task $i \in C$ needs. Worker qualifications are represented by ξ_{kql}, which is a binary parameter equal to 1 if worker k has at least a proficiency level l for skill q. The objective of the WSRP-PL is to minimize the total cost while ensuring that: each task is fulfilled precisely once; the total duration of each route does not exceed the working day duration; and the total walking distance of each team does not exceed the distance limit. Figure 3.1 shows an example of a feasible solution to a toy WSRP-PL instance with nine tasks. In this example, there are four workers (1 to 5) and four potential skills (blue, yellow, green, and red), each with two proficiency levels (dark and light tones). To complete the tasks, the workers are divided into two teams. The green team is formed by workers 1 and 2 while the blue team is formed by workers 4 and 5. Worker 3 is not assigned to a team and remains at the depot for the day. The green team completes tasks 1, 2, and 4; the blue team completes tasks 5 to 9; and task 3 is outsourced.

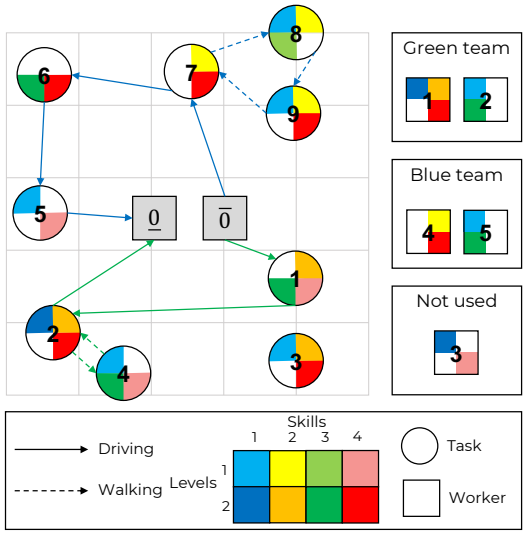


Figure 3.1: WSRP-PL solution example.

To mathematically state the WSRP-PL, we start by defining an additional set $\mathcal{N}^+ = \{n+1, n+2, \ldots, 2n\} \cup \left\{\overline{0}^*\right\}$ that contains a copy of every node $i \in \mathcal{N} \setminus \{\underline{0}\}$. Any node $(i+n) \in \mathcal{N}^+$ represents the end of a subtour starting at node $i \in C$. Node $\overline{0}^*$ represents the end of a subtour starting at $\overline{0}$. Let $d(i)$ be the copy of node $i \in \mathcal{N} \setminus \{\underline{0}\}$. We

also define two sets of arcs. Arc set $\mathcal{A}^1 = \left\{(i,j) : i \in (\mathcal{N} \cup \mathcal{N}^+) \setminus \left\{\underline{0}\right\}, j \in \mathcal{N}\right\}$ contains the arcs in which teams can drive. Arc set $\mathcal{A}^2 = \left\{(i,j) : i \in \mathcal{N} \setminus \left\{\underline{0}\right\}, j \in (\mathcal{N} \cup \mathcal{N}^+) \setminus \left\{\overline{0}, \underline{0}\right\} | \delta_{ij}\right.$ contains the arcs in which teams can walk. The reader should note that some arcs belong to both $\mathcal{A}^1$ and $\mathcal{A}^2$. Building on top of the formulations presented by Kovacs et al. (2012) and Cabrera et al. (2022), the arc-based integer programming formulation (AF) of the WSRP-PL uses the following binary variables:

- $x_{ij}^t = 1$ if team $t \in \mathcal{T}$ drives from node i to node j, for $(i,j) \in \mathcal{A}^1$ and 0, otherwise,
- $h_{ij}^t = 1$ if team $t \in \mathcal{T}$ walks from node i to node j, for $(i,j) \in \mathcal{A}^2$ and 0, otherwise,
- $g_i^t = 1$ if team $t \in \mathcal{T}$ parks the vehicle at node $i \in \mathcal{N} \setminus \left\{\underline{0}\right\}$ and 0, otherwise,
- $y_i^t = 1$ if team $t \in \mathcal{T}$ is assigned to complete task $i \in \mathcal{C}$ and 0, otherwise,
- $v_k^t = 1$ if worker $k \in \mathcal{W}$ is assigned to team $t \in \mathcal{T}$ and 0, otherwise,
- $z_i = 1$ if task $i \in \mathcal{C}$ is outsourced and 0, otherwise,
- $w_t = 1$ if team $t \in \mathcal{T}$ is selected and 0, otherwise.

We also define the following continuous variables:

- u_i^t is the arrival time of team $t \in \mathcal{T}$ to node $i \in \mathcal{N} \cup \mathcal{N}^+$.

An arc-based formulation for the WSRP-PL can be stated as follows:

$$\min \sum_{(i,j) \in \mathcal{A}^1} \sum_{t \in \mathcal{T}} \mu_{ij} c^v x_{ij}^t + \sum_{i \in \mathcal{C}} f_i z_i \tag{3.1}$$

subject to

$$\sum_{t \in \mathcal{T}} v_k^t \leq 1 \qquad \forall k \in \mathcal{W} \tag{3.2}$$

$$\sum_{k \in \mathcal{W}} v_k^t \geq w_t \qquad \forall t \in \mathcal{T} \tag{3.3}$$

$$\sum_{k \in \mathcal{W}} v_k^t \leq \omega w_t \qquad \forall t \in \mathcal{T} \tag{3.4}$$

$$w_t \geq w_{t+1} \qquad \forall t \in \mathcal{T} | t < |\mathcal{T}| \tag{3.5}$$

$$w_t \leq \sum_{i \in \mathcal{C}} y_i^t \qquad \forall t \in \mathcal{T} \tag{3.6}$$

$$\sum_{t \in \mathcal{T}} y_i^t + z_i = 1 \qquad \forall i \in \mathcal{C} \tag{3.7}$$

$$\sum_{(\overline{0},j) \in \mathcal{A}^1} x_{\overline{0}j}^t + \sum_{(\overline{0},j) \in \mathcal{A}^2} h_{\overline{0}j}^t = w_t \qquad \forall t \in \mathcal{T} \tag{3.8}$$

$$\sum_{(i,\underline{0}) \in \mathcal{A}^1} x_{i\underline{0}}^t = w_t \qquad \forall t \in \mathcal{T} \tag{3.9}$$

$$\sum_{(i,j) \in \mathcal{A}^1} x_{ij}^t + \sum_{(i,j) \in \mathcal{A}^2} h_{ij}^t = y_j^t \qquad \forall j \in \mathcal{C}, t \in \mathcal{T} \tag{3.10}$$

$$\sum_{(j,d(i)) \in \mathcal{A}^2} h_{jd(i)}^t = g_i^t \qquad \forall i \in \mathcal{N} \setminus \left\{\underline{0}\right\}, t \in \mathcal{T} \tag{3.11}$$

$$\sum_{(i,j) \in \mathcal{A}^2} h_{ij}^t - \sum_{(j,i) \in \mathcal{A}^2} h_{ji}^t = g_i^t \qquad \forall i \in \mathcal{N} \setminus \left\{\underline{0}\right\}, t \in \mathcal{T} \tag{3.12}$$

$$\begin{aligned} &\sum_{(j,i) \in \mathcal{A}^2} h_{ji}^t + \sum_{(j,i) \in \mathcal{A}^1} x_{ji}^t \\ &- \sum_{(i,j) \in \mathcal{A}^2} h_{ij}^t - \sum_{(i,j) \in \mathcal{A}^1} x_{ij}^t = 0 \end{aligned} \qquad \forall j \in \mathcal{N} \cup \mathcal{N}^+, t \in \mathcal{T} \tag{3.13}$$

$$\sum_{(j,i) \in \mathcal{A}^2} h_{ji}^t + \sum_{(j,i) \in \mathcal{A}^1} x_{ji}^t \leq w_t \qquad \forall j \in \mathcal{N} \cup \mathcal{N}^+, t \in \mathcal{T} \tag{3.14}$$

$$\sum_{t \in \mathcal{T}} (h_{ij}^t + x_{ij}^t) \leq 1 \qquad \forall (i,j) \in \mathcal{A}^1 \cap \mathcal{A}^2 \tag{3.15}$$

$$\begin{aligned} &\tau_{ij} x_{ij}^t + \eta_{ij} h_{ij}^t + s_i y_i^t + u_i^t \\ &\quad -\phi(1 - h_{ij}^t - x_{ij}^t) \leq u_j^t \end{aligned} \qquad \forall (i,j) \in \mathcal{A}^1 \cap \mathcal{A}^2, t \in \mathcal{T} \tag{3.16}$$

$$\begin{aligned} &\tau_{ij} x_{ij}^t + s_i y_i^t + u_i^t \\ &\quad -\phi(1 - x_{ij}^t) \leq u_j^t \\ &\eta_{ij} h_{ij}^t + s_i y_i^t + u_i^t \end{aligned} \qquad \forall (i,j) \in \mathcal{A}^1 \setminus (\mathcal{A}^1 \cap \mathcal{A}^2), t \in \mathcal{T} \tag{3.17}$$

$$-\phi(1-h_{ij}^t) \leq u_j^t \qquad \forall (i,j) \in \mathcal{A}^2 \setminus (\mathcal{A}^1 \cap \mathcal{A}^2), t \in \mathcal{T} \tag{3.18}$$

$$u_i^t \leq u_{d(i)}^t \qquad \forall i \in \mathcal{N} \setminus \{\underline{0}\}, t \in \mathcal{T} \tag{3.19}$$

$$u_i^t \leq b_i \qquad \forall i \in \mathcal{C}, t \in \mathcal{T} \tag{3.20}$$

$$u_i^t \geq a_i \qquad \forall i \in \mathcal{C}, t \in \mathcal{T} \tag{3.21}$$

$$\sum_{(i,j)\in\mathcal{A}^2} \delta_{ij} h_{ij}^t \leq \zeta \qquad \forall t \in \mathcal{T} \tag{3.22}$$

$$\sum_{k\in\mathcal{W}} \xi_{kql} v_k^t \geq \nu_{iql} y_i^t \qquad \forall i \in \mathcal{C}, t \in \mathcal{T}, q \in \mathcal{Q}, l \in \mathcal{L} \tag{3.23}$$

$$x_{ij}^t \in \{0,1\} \qquad \forall t \in \mathcal{T}, (i,j) \in \mathcal{A}^1 \tag{3.24}$$

$$h_{ij}^t \in \{0,1\} \qquad \forall t \in \mathcal{T}, (i,j) \in \mathcal{A}^2 \tag{3.25}$$

$$g_i^t \in \{0,1\} \qquad \forall t \in \mathcal{T}, i \in \mathcal{N} \setminus \{\underline{0}\} \tag{3.26}$$

$$y_i^t \in \{0,1\} \qquad \forall t \in \mathcal{T}, i \in \mathcal{C} \tag{3.27}$$

$$v_k^t \in \{0,1\} \qquad \forall t \in \mathcal{T}, k \in \mathcal{W} \tag{3.28}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{C} \tag{3.29}$$

$$w_t \in \{0,1\} \qquad \forall t \in \mathcal{T} \tag{3.30}$$

$$u_i^t \geq 0 \qquad \forall t \in \mathcal{T}, i \in \mathcal{N} \cup \mathcal{N}^+. \tag{3.31}$$

The objective function (3.1) minimizes the total cost comprised of the routing and outsourcing costs. Constraints (3.2) ensure that a worker is only assigned to one team. Constraints (3.3) and (3.4) impose a minimum and a maximum number of workers for each selected team. Constraints (3.5) remove symmetric solutions with respect to the selection of teams. Constraints (3.6) state that every selected team must complete at least one task. Constraints (3.7) ensure that all tasks are either completed or outsourced. Constraints (3.8) state that all teams must leave the start depot. Constraints (3.9) ensure that all the teams arrive at the end depot.

Constraints (3.10) guarantee that all the tasks are executed by a team. Constraints (3.11) ensure that the vehicle is recovered after performing a subtour. Constraints (3.12) and (3.13) ensure flow conservation. Constraints (3.14) impose a

limit on the number of arcs that can be used by a team departing from any location. Constraints (3.15) state that an arc can only be used once. Constraints (3.16)-(3.19) define the time of arrival at every location. Constraints (3.20)-(3.21) ensure that the completion of each task starts within their time windows. Constraints (3.22) impose a limit on the duration the overall walking distance. Constraints (3.23) ensure that tasks are fulfilled by a team with the appropriate skills.

The proposed arc-based formulation is compact (i.e., the number of variables and constraints is polynomial with respect to the dimension of a problem instance). However, as mentioned by Dellaert et al. (2019), arc-based formulations suffer from poor performance caused by the bad quality of the lower bound obtained by solving its LP relaxation. As a result, we propose using a new path-based formulation which is the topic of the next section.

3.3 Path-based formulation

Let $\bar{\mathcal{G}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ be a directed graph, henceforth referred to as the *modified network*, where $\bar{\mathcal{N}}$ is the set of nodes and $\bar{\mathcal{A}}$ is the set of directed arcs. The set of nodes $\bar{\mathcal{N}} = \mathcal{N} \cup \{\mathfrak{s}\} \cup \mathcal{W}$ comprises the nodes in graph $\mathcal{G}$, a source node $\mathfrak{s}$, and one node for every worker $k \in \mathcal{W}$. There are two types of arcs in the modified network, namely, routing and scheduling arcs. Routing arcs in $\mathcal{A}$ represent connections between tasks, as defined in Section 3.2. Scheduling arcs represent the selection of workers. More precisely, let $\mathcal{A}' = \mathcal{A}'_1 \cup \mathcal{A}'_2 \cup \mathcal{A}'_3$ be the scheduling arcs, where $\mathcal{A}'_1 = \{(\mathfrak{s}, k) : k \in \mathcal{W}\}$ are arcs from the source node to every worker node, $\mathcal{A}'_2 = \left\{(k, \overline{0}) : k \in \mathcal{W}\right\}$ are arcs from every worker node to the start depot node, and $\mathcal{A}'_3 = \{(k, k') : k, k' \in \mathcal{W} | k' > k\}$ are arcs between workers nodes. A worker $k \in \mathcal{W}$ is said to be selected if an arc ending at the corresponding node is used. Figure 3.2 shows an illustrative example of the modified network on an instance that includes five workers and four tasks. The maximum number of workers in a team is three. The number of skills is set to four and each skill has two proficiency levels. For the

sake of simplicity, some routing arcs were omitted. Note that by using this graph, both scheduling and routing decisions can be made simultaneously.

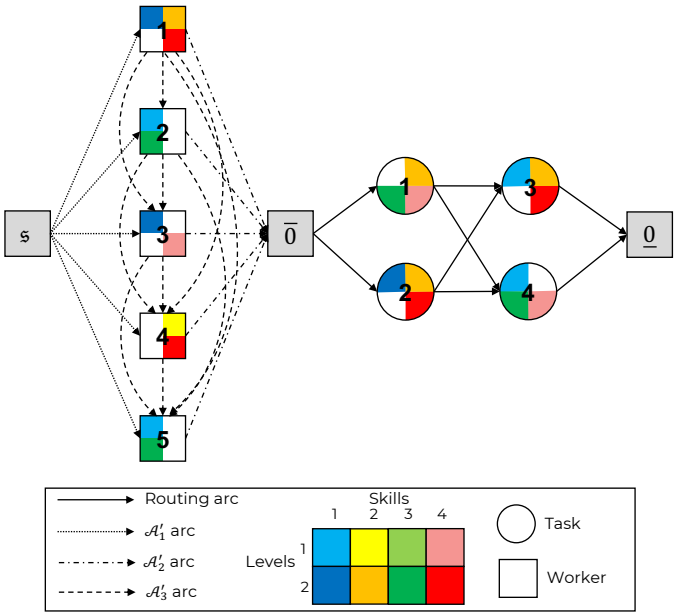


Figure 3.2: Modified network example.

Let path p be an ordered set of directed arcs starting at the source node $\mathfrak{s}$ and ending at the end depot $\underline{0}$. Let also $\mathcal{P}$ be the set of all feasible pseudo-elementary paths. Pseudo-elementary means that each node included in the path can only appear once unless it is used as a parking spot. The scheduling arcs in p are contained in subsets $\mathcal{A}_p^s$. The driving and the walking arcs in p are contained in subsets $\mathcal{A}_p^d$ and $\mathcal{A}_p^w$, respectively. Let $\mathcal{W}_p$ be the workers selected in the path. Also, let $\mathcal{C}_p$ be the tasks completed in the path starting the service within their corresponding time windows. A path p is feasible if the following conditions hold:

$$|\mathcal{A}_p^s| \leq \omega + 1 \tag{3.32}$$

$$\sum_{(i,j)\in\mathcal{A}_p^w} \eta_{ij} + \sum_{(i,j)\in\mathcal{A}_p^d} \tau_{ij} + \sum_{i\in\mathcal{C}_p} s_i \leq \phi \tag{3.33}$$

$$\sum_{(i,j)\in\mathcal{A}_p^w} \delta_{ij} \leq \zeta \tag{3.34}$$

$$\sum_{k \in \mathcal{W}_p} \xi_{kql} \geq \nu_{iql} \qquad \forall i \in \mathcal{C}_p, q \in \mathcal{Q}, l \in \mathcal{L}. \tag{3.35}$$

Condition (3.32) ensures that the maximum number of workers per team is respected. Condition (3.33) states that the total duration of a path must be less than the time limit. Condition (3.34) guarantees that the total walking distance respects the limit. Conditions (3.35) state that all tasks completed in the path must be fulfilled by a team with the appropriate skills. Figure 3.3 shows an example of a path carried out by a team composed of workers 1 and 2. This team fulfills tasks 1, 2, and 4 (in that order).

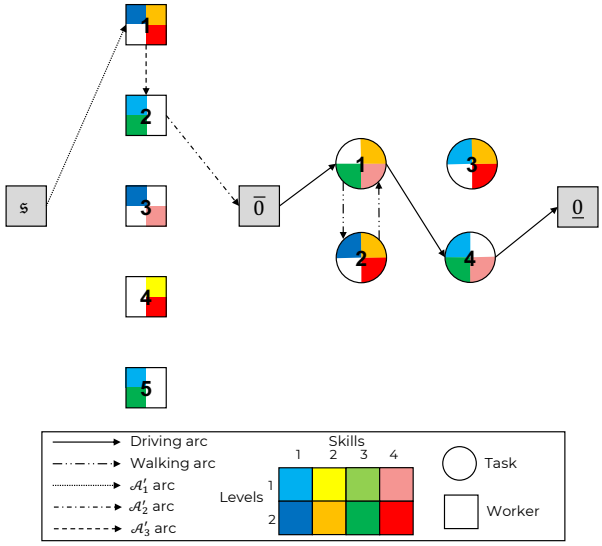


Figure 3.3: Feasible path example.

The reader should note that the path shown in Figure 3.3 could alternatively use worker 5 instead of worker 2. More generally, consider the case in which two workers k and k' have the same qualifications $\xi_{kql} = \xi_{k'ql}$ for every skill $q \in \mathcal{Q}$ and proficiency level $l \in \mathcal{L}$. Then, every path $p \in \mathcal{P}$ that uses worker k can be replicated by using worker k'. To deal with this potential source of inefficiency due to symmetry, we slightly modify graph $\bar{\mathcal{G}}$ as follows. First, we define a set of worker profiles $\mathcal{O}$. All the workers with the same level of proficiency at every skill are associated with one unique worker profile. Let Γ_o be the number of available workers with profile

$o \in \mathcal{O}$. Then, we can re-define node set $\bar{\mathcal{N}}$ as $\bar{\mathcal{N}} = \mathcal{N} \cup \{\mathfrak{s}\} \cup \mathcal{O}$, in which instead of having one node for every worker, the graph has one node per worker profile. Also, we re-define the set of scheduling arcs $\mathcal{A}'$ as $\mathcal{A}' = \mathcal{A}'_1 \cup \mathcal{A}'_2 \cup \mathcal{A}'_3$, where $\mathcal{A}'_1 = \{(\mathfrak{s}, o)^\alpha : o \in \mathcal{O}, \alpha \in 1 \dots \Gamma_o | \alpha \leq \omega\}$ are arcs from the source to every worker profile node, $\mathcal{A}'_2 = \left\{(o, \bar{0}) : o \in \mathcal{O}\right\}$ are arcs from every profile node to the start depot node, and $\mathcal{A}'_3 = \{(o, o')^\alpha : o, o' \in \mathcal{O}, \alpha \in 1 \dots \Gamma_{o'} | o' > o \wedge \alpha \leq \omega\}$. A worker profile $o \in \mathcal{O}$ is said to be selected if an arc $(\cdot, o)^\alpha$ ending at the corresponding node is used. The number of workers used of the associated worker profile is α. For ease of notation we will denote the number of workers selected when using arc $(i, j) \in \mathcal{A}'$ by χ_{ij}. Similarly, we denote their profile $o \in \mathcal{O}$ by ϱ_{ij}.

Figure 3.4 shows how the graph presented in Figure 3.2 can be represented using the updated modified network. Numbers next to the arcs represent the number of workers selected if the arc is used. By convention, if no number is shown along the arc, the number of workers is 1. Note that workers 2 and 5 are now represented by a single node. Moreover, note that there are two arcs from the source node to the node corresponding to profile 2. Similarly, there are two arcs between the nodes representing profile 1 and profile 2. As a result, this alternative representation allows us to remove redundant paths from the graph.

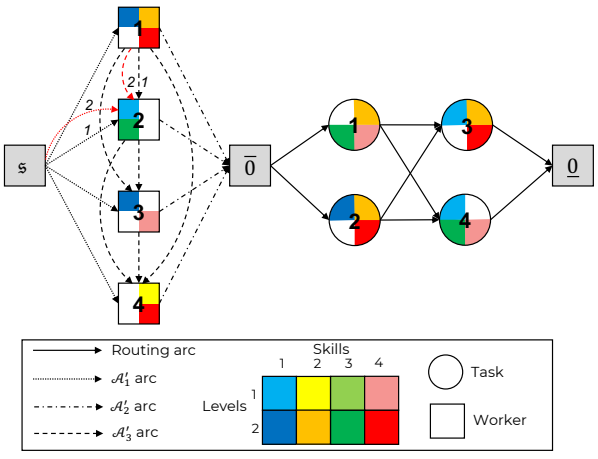


Figure 3.4: Modified network example using worker profiles.

The cost c_p of a path is equal to the driving cost, that is,

$$c_p = \sum_{(i,j)\in\mathcal{A}_p^d} \mu_{ij} c^v. \tag{3.36}$$

Let a_{ip} be a binary parameter that takes the value 1 if and only if path $p \in \mathcal{P}$ completes task $i \in \mathcal{C}$, and let b_{op} be a parameter that takes the value of the number of workers of profile $o \in \mathcal{O}$ in the team assigned to path $p \in \mathcal{P}$. Also, let ϑ_i be a binary variable equal to 1 if task $i \in \mathcal{C}$ is outsourced and 0 otherwise. Finally, let λ_p be a binary variable equal to 1 if path $p \in \mathcal{P}$ is selected and 0 otherwise. A set partitioning (SP) formulation for the WSRP-PL can be stated as follows:

$$\min \sum_{p\in\mathcal{P}} \lambda_p c_p + \sum_{i\in\mathcal{C}} \vartheta_i f_i \tag{3.37}$$

subject to

$$\sum_{p\in\mathcal{P}} a_{ip}\lambda_p + \vartheta_i = 1 \qquad \forall i \in \mathcal{C} \tag{3.38}$$

$$\sum_{p\in\mathcal{P}} b_{op}\lambda_p \leq \Gamma_o \qquad \forall o \in \mathcal{O} \tag{3.39}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{3.40}$$

$$\vartheta_i \in \{0,1\} \qquad \forall i \in \mathcal{C}. \tag{3.41}$$

The objective function (3.37) minimizes the total cost. Constraints (3.38) ensure that all tasks are fulfilled or outsourced. Constraints (3.39) guarantee that the number of workers used of each profile is less or equal than the number available. The SP model has one variable for every possible feasible path. However, the number of feasible paths $|\mathcal{P}|$ grows exponentially with the number of tasks and workers. As a result, enumerating all the feasible paths to solve SP is only possible for trivial instances. As an alternative, the SP can be solved using a branch-price-and-cut algorithm, which is described next.

3.4 Solution method

We propose a BPC algorithm for the above SP formulation. BPC is a branch-and-bound procedure for solving mixed integer linear programming models with a large number of variables. The algorithm's objective is to find the optimal solution without explicitly enumerating the complete set of feasible variables (paths). Instead, the algorithm iteratively builds a subset $\overline{\mathcal{P}} \subseteq \mathcal{P}$ of promising paths. To do so, the algorithm relies on two optimization problems. A *master problem* that evaluates the performance of the current subset of paths and a *pricing problem* that finds new promising paths. Usually, the subset $\overline{\mathcal{P}}$ is initialized using heuristic methods. We refer the interested reader to Costa et al. (2019b) for a review of techniques and applications of BPC algorithms.

In our case, the master problem is the relaxed version of formulation (3.37)-(3.41). The master problem is obtained by relaxing the integrality constraints on the λ_p and ϑ_i variables. Let $\pi_i \in \mathbb{R}$ and $\sigma_o \leq 0$ be the dual variables associated with constraints (3.38) and (3.39), respectively. Then, the pricing problem, which aims to find the pseudo-elementary path in graph $\bar{\mathcal{G}}$ that minimizes the total reduced cost is formulated as follows:

$$\min_{p \in \mathcal{P}} \left\{ \mathfrak{r}_p = c_p - \sum_{i \in C} a_{ip} \pi_i - \sum_{o \in O} b_{op} \sigma_o \right\}. \tag{3.42}$$

If the pricing problem finds a path with $\mathfrak{r}_p < 0$, we add the corresponding path p to the subset $\overline{\mathcal{P}}$. In the opposite case, the current master problem is solved optimally. If the optimal solution to the master problem is fractional, the algorithm uses a *cut separator* to generate valid inequalities and strengthen the linear relaxation. Lastly, if the solution is still fractional, the algorithm resorts to branching. In the following subsections, we present a detailed description of the algorithm used to solve the pricing problem, the cut separator, and the branching strategy. We also describe several ideas that we use to speed up our BPC algorithm.

3.4.1 Pricing problem

The goal of the pricing problem is to find the path with the lowest reduced cost. The total reduced cost of a path corresponds to the sum of the reduced cost of all its arcs. The reduced cost r_{ij} of an arc $(i,j) \in \bar{\mathcal{A}}$ depends on the arc's type (i.e., scheduling or routing). The reduced cost of a scheduling arc $(i,j) \in \mathcal{A}'$ is defined as:

$$r_{ij} = \begin{cases} -\chi_{ij}\sigma_{\varrho_{ij}}, & (i,j) \in \mathcal{A}'_1 \cup \mathcal{A}'_3; \\ 0, & (i,j) \in \mathcal{A}'_2. \end{cases} \tag{3.43}$$

The reduced cost of a routing arc $(i,j) \in \mathcal{A}$ depends on the transportation mode used. In particular, if arc (i,j) is traversed on foot, the reduced cost r^w_{ij} is defined as:

$$r^w_{ij} = \begin{cases} -\pi_j, & (i,j) \in \mathcal{A} | j \neq \overline{0}; \\ 0, & (i,j) \in \mathcal{A} | j = \overline{0}. \end{cases} \tag{3.44}$$

Conversely, if arc $(i,j) \in \mathcal{A}$ is traversed by driving, the reduced cost r^d_{ij} is defined as:

$$r^d_{ij} = \begin{cases} \mu_{ij}c^v - \pi_j, & (i,j) \in \mathcal{A} | j \neq \underline{0}; \\ \mu_{ij}c^v, & (i,j) \in \mathcal{A} | j = \underline{0}. \end{cases} \tag{3.45}$$

To identify the path with the lowest reduced cost, an (elementary) shortest path problem with resource constraints and park-and-loop (ESPPRC-PL) from the source node $\mathfrak{s}$ to the end depot $\underline{0}$ has to be solved. A distinctive feature of this pricing problem is that paths have a park-and-loop structure. Also, paths are constrained

by three resources: the time ϕ, the walking distance ζ, and the size of the team ω. The ESPPRC is typically solved by means of labeling algorithms, where labels represent partial paths (Feillet et al. 2004; Parragh and Cordeau 2017; Pessoa et al. 2018). The efficiency of these algorithms depends on the ability to identify and discard dominated partial paths by performing dominance checks (Schrotenboer et al. 2019). In the case of the ESPPRC-PL, two labels at a task node are only comparable if they have the same parking spot, which may lessen the effectiveness of the dominance checks. As a result, the number of labels stored can grow rapidly. As an alternative, the ESPPRC-PL can be solved using the pulse algorithm (PA). The PA is a recursive algorithm that performs a depth-first search exploration of the network. Crucially, the PA does not store labels at any task node and does not rely on asserting dominance between them. Instead, to avoid enumerating all the possible paths in the graph, the algorithm uses pruning strategies that aggressively and effectively discard partial paths. This algorithm was originally proposed by Lozano et al. (2016) and was later extended by Cabrera et al. (2023) to handle the park-and-loop structure of the paths. Building on these previous works, we extend the PA to consider the scheduling decisions that arise in the WSRP-PL.

Algorithm 6 presents the main logic of the PA. Line 1 creates an empty partial path. Lines 2 to 4 initialize the reduced cost, the team size, and a $|\mathcal{Q}| \times |\mathcal{L}|$ matrix that contains the number of workers that have at least a proficiency level $l \in \mathcal{L}$ for skill $q \in \mathcal{Q}$. We will refer to this matrix, as the skills matrix. Line 5 extends the partial path at the source node $\mathfrak{s}$. Finally, line 6 returns the path with the lowest reduced cost.

Algorithm 7 defines the `pulse_scheduling` function, where $\Pi^+(i)$ contains all the scheduling arcs leaving node i. Line 1 adds the current node to the partial path. From lines 2 to 16, the algorithm recursively propagates the pulse through arc $(i, j) \in \Pi^+(i)$. Line 3 updates the cumulative reduced cost. Line 4 checks if the current arc arrives at the start depot $\overline{0}$. If so, line 5 temporarily removes all the nodes associated with tasks that cannot be performed by the current team. That

Algorithm 6 `pulseSearch` function

Require: $\bar{\mathcal{G}}$, directed graph; ϕ, duration limit; ζ, walking distance limit; ω, team size limit; $\mathfrak{s}$, start node; $\underline{0}$, end node; Δ, bound step size; $[\bar{t}, \underline{t}]$, bounding time limits.

Ensure: $\mathscr{P}^*$, optimal path.

1: $\mathscr{P} \leftarrow \emptyset$
2: $r(\mathscr{P}) \leftarrow 0$
3: $n(\mathscr{P}) \leftarrow 0$
4: $S_{ql}(\mathscr{P}) \leftarrow 0$
5: `pulse_scheduling`$(\mathfrak{s}, r(\mathscr{P}), S_{ql}(\mathscr{P}), n(\mathscr{P}), \mathscr{P})$ ▷ see *Algorithm 7*
6: **return** $\mathscr{P}^*$

is, the tasks $i' \in C$ for which the number of workers S_{ql} with proficiency level l at skill q is lower than the number required $\nu_{i'ql}$. Then, line 6 runs the bounding procedure given the bound step size Δ and the bounding time limits $[\bar{t}, \underline{t}]$. Finally, line 7 extends a pulse at the start depot using Algorithm 8. If the current arc arrives at a different node, line 9 updates the skills matrix associated with the current team of workers and line 10 updates the team size. Then, line 11 checks if the team size is lower than the limit. If so, the algorithm evaluates if the current team of workers is dominated. If the team is not dominated, then the algorithm recursively propagates the pulse to node j.

Algorithm 8 defines the `pulse_routing` function, where $\Sigma^+(i)$ contains all the routing arcs leaving node i. Lines 1 to 3 try to prune the current partial path using the feasibility, bounds, and rollback pruning strategies. If the partial path is not pruned, line 4 adds the current node to the partial path. From lines 5 to 9, the algorithm recursively propagates the pulse by driving through arc $(i, j) \in \Sigma^+(i)$. From lines 10 to 14, the algorithm recursively propagates the pulse using Algorithm 3 by walking to node j. When this occurs, the vehicle is parked at node i.

Algorithm 9 defines the `pulse_parked` function. Lines 1 to 3 try to prune the current partial path using the feasibility, bounds, and rollback pruning strategies. If the partial path is not pruned, line 4 adds the current node to the partial path. From lines 5 to 18, the algorithm recursively propagates the pulse by walking through arc

Algorithm 7 `pulse_scheduling` function

Require: i, current node; r, cumulative reduced cost; S_{ql}, current skills matrix; n, cumulative team size; $\mathscr{P}$, partial path.

1: $\mathscr{P}' \leftarrow \mathscr{P} \cup \{i\}$
2: **for** $j \in \Pi^+(i)$ **do**
3: $\quad r(\mathscr{P}') \leftarrow r(\mathscr{P}) + r_{ij}$
4: $\quad$ **if** $j = \overline{0}$ **then**
5: $\quad\quad$ `prune_by_skills`$(C, S_{ql}(\mathscr{P}'))$
6: $\quad\quad$ `start_bounding`$(\bar{\mathcal{G}}, \Delta, [\bar{t}, \underline{t}])$ ▷ see *§3.4.2*
7: $\quad\quad$ `pulse_routing`$(j, r(\mathscr{P}'), t(\mathscr{P}'), w(\mathscr{P}'), \mathscr{P}')$ ▷ see *Algorithm 8*
8: $\quad$ **else if** $j \neq \overline{0}$ **then**
9: $\quad\quad$ $S_{ql}(\mathscr{P}') \leftarrow$ `update_matrix`$(S_{ql}(\mathscr{P}), \varrho_{ij}, \chi_{ij})$
10: $\quad\quad$ $n(\mathscr{P}') \leftarrow n(\mathscr{P}) + \chi_{ij}$
11: $\quad\quad$ **if** $n(\mathscr{P}') \leq \omega$ **then**
12: $\quad\quad\quad$ **if** $\neg$`dominance`$(j, r(\mathscr{P}'), S_{ql}(\mathscr{P}'))$ **then** ▷ see *§3.4.2*
13: $\quad\quad\quad\quad$ `pulse_scheduling`$(j, r(\mathscr{P}'), S_{ql}(\mathscr{P}'), n(\mathscr{P}'), \mathscr{P}')$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end if**
16: $\quad$ **end if**
17: **end for**
18: **return** `void`

$(i, j) \in \Sigma^+(i)$. At line 8, the algorithm checks if the pulse is returning to the parking spot. If so, at line 13 the algorithm recursively propagates the pulse using Algorithm 8.

3.4.2 Pruning strategies

This section outlines the core pruning strategies implemented for the ESPPRC-PL adapted from Lozano and Medaglia (2013) and Lozano et al. (2016). Section 3.4.2 provides details of line 12 in Algorithm 7. Sections 3.4.2, 3.4.2, and 3.4.2, provide details of lines 1, 2, and 3 in Algorithm 8, respectively.

Dominance pruning

During the recursive search, a worker node $o \in O$ can be reached more than one time by different pulses representing different teams of workers. With this in mind,

Algorithm 8 `pulse_routing` function

Require: i, current node; r, cumulative reduced cost; t, cumulative time; w, cumulative walking distance; $\mathscr{P}$, partial path.

if $\neg$`feasibility`$(i, t(\mathscr{P}), w(\mathscr{P}))$ **then** ▷ see *§3.4.2*
  if $\neg$`bounds`$(i, r(\mathscr{P}), t(\mathscr{P}))$ **then** ▷ see *§3.4.2*
    if $\neg$`rollback`$(i, r(\mathscr{P}), t(\mathscr{P}), w(\mathscr{P}), \mathscr{P})$ **then** ▷ see *§3.4.2*
      $\mathscr{P}' \leftarrow \mathscr{P} \cup \{i\}$
      for $j \in \Sigma^+(i)$ **do** ▷ Driving
        $r(\mathscr{P}') \leftarrow r(\mathscr{P}) + r^d_{ij}$
        $t(\mathscr{P}') \leftarrow \max\{a_j, t(\mathscr{P}) + \tau_{ij} + s_j\}$
        `pulse_routing`$(j, r(\mathscr{P}'), t(\mathscr{P}'), w(\mathscr{P}'), \mathscr{P}')$
      end for
      for $j \in \Sigma^+(i)$ **do** ▷ Walking
        $r(\mathscr{P}') \leftarrow r(\mathscr{P}) + r^w_{ij}$
        $t(\mathscr{P}') \leftarrow \max\{a_j, t(\mathscr{P}) + \eta_{ij} + s_j\}$
        $w(\mathscr{P}') \leftarrow w(\mathscr{P}) + \delta_{ij}$
        `pulse_parked`$(j, r(\mathscr{P}'), t(\mathscr{P}'), w(\mathscr{P}'), i, \mathscr{P}')$ ▷ see *Algorithm 9*
      end for
    end if
  end if
end if
return `void`

we define dominance relations between two partial paths $\mathscr{P}_1$ and $\mathscr{P}_2$ arriving at node o. Partial path $\mathscr{P}_1$ *dominates* $\mathscr{P}_2$ if:

$$r(\mathscr{P}_1) < r(\mathscr{P}_2) \quad \wedge \quad n(\mathscr{P}_1) < n(\mathscr{P}_2) \quad \wedge \quad S_{ql}(\mathscr{P}_1) > S_{ql}(\mathscr{P}_2) \quad \forall q \in \mathcal{Q}, l \in \mathcal{L}. \tag{3.46}$$

Condition (3.46) states that a partial path $\mathscr{P}_1$ dominates $\mathscr{P}_2$ if it has a lower reduced cost, if it includes fewer workers, and if it includes a higher number of workers with at least a level of proficiency $l \in \mathcal{L}$ at every skill $q \in \mathcal{Q}$. This condition guarantees that every path that starts with partial path $\mathscr{P}_2$ can be replicated using partial path $\mathscr{P}_1$. To check conditions (3.46) we store a list of nondominanted labels $\mathscr{M}_o = \left\{(r(\mathscr{P}_m), S_{ql}(\mathscr{P}_m))|m = 1, \ldots, |\mathscr{M}|\right\}$ corresponding to partial paths at every node $o \in \mathcal{O}$, where $|\mathscr{M}|$ denotes the memory size. Note that the correctness of the algorithm does not rely on storing all non-dominated labels. Moreover, choosing $|\mathscr{M}| << \infty$ poses a trade-off between the ability to efficiently discard unpromising

Algorithm 9 `pulse_parked` function

Require: i, current node; r, cumulative reduced cost; t, cumulative time; w, cumulative walking distance; ps, parking spot; $\mathscr{P}$, partial path.

if ¬`feasibility`$(i, t(\mathscr{P}), w(\mathscr{P}))$ **then** ▷ see *§3.4.2*
  if ¬`bounds`$(i, r(\mathscr{P}), t(\mathscr{P}))$ **then** ▷ see *§3.4.2*
    if ¬`rollback`$(i, r(\mathscr{P}), t(\mathscr{P}), w(\mathscr{P}), \mathscr{P})$ **then** ▷ see *§3.4.2*
      $\mathscr{P}' \leftarrow \mathscr{P} \cup \{i\}$
      for $j \in \Sigma^+(i)$ **do**
        $r(\mathscr{P}') \leftarrow r(\mathscr{P}) + r_{ij}^w$
        $w(\mathscr{P}') \leftarrow w(\mathscr{P}) + \delta_{ij}$
        if $ps = j$ **then**
          $t(\mathscr{P}') \leftarrow t(\mathscr{P}) + \eta_{ij}$
          if $j \in C$ **then**
            $r(\mathscr{P}') \leftarrow r(\mathscr{P}') + \pi_j$
          end if
          `pulse_routing`$(j, r(\mathscr{P}'), t(\mathscr{P}'), w(\mathscr{P}'), \mathscr{P}')$ ▷ see *Algorithm 8*
        else if $ps \neq j$ **then**
          $t(\mathscr{P}') \leftarrow \max\{a_j, t(\mathscr{P}) + \eta_{ij} + s_j\}$
          `pulse_parked`$(j, r(\mathscr{P}'), t(\mathscr{P}'), w(\mathscr{P}'), ps, \mathscr{P}')$
        end if
      end for
    end if
  end if
end if
return `void`

teams of workers and the amount of memory required to execute the algorithm. In our algorithm we set $|\mathscr{M}| = 3$.

Infeasibility pruning

The purpose of the infeasibility pruning strategy is to prune partial paths that exceed the resource constraints. More specifically, a partial path $\mathscr{P}$ from $\mathfrak{s}$ to $j \in C$ is pruned if any of the following condition holds:

$$t(\mathscr{P}) > \phi; \tag{3.47}$$

$$w(\mathscr{P}) > \zeta; \tag{3.48}$$

$$t(\mathscr{P}) > b_j. \tag{3.49}$$

Condition (3.47) states that a partial path is pruned if the total time of the partial path is greater than the time limit. Similarly, condition (3.48) states that a partial path is pruned if the cumulative walking distance if greater than the limit. Finally, condition (3.49) states that the partial path is pruned if node j is visited after the latest starting time of the corresponding task.

Bounds pruning

The previous strategy prunes partial paths as soon as it becomes evident that they will not meet the resource constraints. This strategy may be very effective when solving tight instances. However, it may not be enough if the available quantity for each resource is large. To address this, the bounds pruning strategy prunes partial paths as soon as there is enough information to prove that they will not improve the current best solution in terms of the objective function $r(\mathscr{P}^*)$. With this objective in mind, we can prune a partial path if the following condition holds:

$$r(\mathscr{P}) + \underline{r}(j, t(\mathscr{P})) \geq r(\mathscr{P}^*), \tag{3.50}$$

where $\underline{r}(j, t(\mathscr{P}))$ is a lower bound on the minimum reduced cost from any node $j \in C$ to the end node $\underline{0}$. In the original PA, these lower bounds are computed before the start of the algorithm by solving an ESPPRC-PL from every node $i \in C$ to the end node $\underline{0}$ while setting the time consumption $t(\mathscr{P})$ to $\left\{\bar{t} - \Delta, \bar{t} - 2\Delta, \bar{t} - 3\Delta, \ldots, \underline{t}\right\}$. As a result, after solving this sequence of problems, the algorithm has a valid lower bound on the minimum reduced cost for every node and for every discrete time step between $\underline{t}$ and $\bar{t}$. Note, however, that information regarding the team configuration is ignored during this step. In a one-to-one implementation of the PA, this would lead to weak bounds. Moreover, preliminary experiments showed that the computational time needed to compute these bounds can be large. Thus, we modified condition (3.50) as follows:

$$r(\mathscr{P}) + \underline{r}(j, t(\mathscr{P}), S_{ql}(\mathscr{P})) \geq r(\mathscr{P}^*), \tag{3.51}$$

where $\underline{r}(j, t(\mathscr{P}), S(\mathscr{P}))$ is a lower bound on the minimum reduced cost from any node $j \in C$ to the end node $\underline{0}$ given the current cumulative resource consumption $t(\mathscr{P})$ and the skills matrix $S_{ql}(\mathscr{P})$ associated with the current team of workers. These bounds are not computed before the start of the algorithm. Instead, they are computed every time that a pulse reaches the start depot $\overline{0}$ following a two step procedure (i.e., lines 5 and 6 of Algorithm 7). In the first step, the algorithm temporarily removes tasks from graph $\bar{\mathcal{G}}$ if the number of workers in the team $S_{ql}(\mathscr{P})$ with at least proficiency level l at skill q is lower than the number of workers required ν_{jql} to perform them. This significantly quickens the pulse propagation during the second step, in which the algorithm solves a ESPPRC-PL from every node $i \in C$ to the end node $\underline{0}$ while setting the time consumption $t(\mathscr{P})$ to $\left\{\bar{t} - \Delta, \bar{t} - 2\Delta, \bar{t} - 3\Delta, \ldots, \underline{t}\right\}$. In Appendix D, we report on an experiment that empirically shows that condition (3.51) significantly increases the effectiveness of the bounds pruning strategy.

Rollback pruning

As stated previously, the PA performs a depth-first search exploration of the network. In practice, this means that a pulse (i.e., a partial path) only stops propagating if it reaches the end depot or if it is pruned by one of the pruning strategies. In some cases, this behavior may have a negative impact on the algorithm's performance, as it may take a while before the algorithm backtracks and corrects poor decisions made earlier. To deal with this potential source of inefficiency, we use the rollback pruning strategy. This strategy evaluates if a partial path $\mathscr{P}_{sk} = \mathscr{P}_{si} \cup \{j\} \cup \{k\}$ should continue to propagate by comparing it with a partial path that skips node j, that is, $\mathscr{P}'_{sk} = \mathscr{P}_{si} \cup \{k\}$. In particular, the partial path $\mathscr{P}_{sk}$ should be pruned if the following conditions hold:

$$r(\mathscr{P}_{sk}) \geq r(\mathscr{P}'_{sk}); \tag{3.52}$$

$$t(\mathscr{P}_{sk}) \geq t(\mathscr{P}'_{sk}). \tag{3.53}$$

Conditions (3.52)-(3.53) state that a partial path should be pruned if its total reduced cost and its total time are greater or equal to the total reduced cost and the total time of a partial path that skips the penultimate node visited.

3.4.3 Initial set of paths

The master problem is always feasible as it is always possible to outsource all tasks. Thus, it is entirely possible to start the BPC algorithm with an empty set of paths. However, it has been observed by multiple researchers that the performance of BPC algorithms is largely affected by the initial set of variables. Our algorithm follows a two-step approach to generate paths carried out by single-worker teams. First, we run our own implementation of the constructive heuristic proposed by Xie et al. (2017). Second, we apply a modified version of the tabu search metaheuristic proposed by Lozano et al. (2016) that considers the skills required by each task to quickly improve those paths.

3.4.4 Cut separator

Although path-based formulations have the potential to offer tighter lower bounds compared to compact formulations, these bounds may still lack the strength necessary to produce an efficient algorithm (Costa et al. 2019b). Consequently, the master problem is usually reinforced with valid inequalities (cuts) that can significantly decrease the size of the branch-and-price tree. In our BPC algorithm, we use the subset row inequalities. These inequalities were proposed by Jepsen et al. (2008) and have since then become an essential component in BPC algorithms for solving many vehicle routing problems (Contardo and Martinelli 2014; Costa et al. 2019b; Marques et al. 2020). Given a subset of tasks $\mathcal{S} \subseteq \mathcal{C}$ of size 3, the subset row inequalities can be defined as follows:

$$\sum_{p \in \mathcal{P}} \left\lfloor 1/2 \sum_{i \in \mathcal{S}} a_{ip} \right\rfloor \lambda_p \leq 1, \qquad \forall \mathcal{S} \subseteq \mathcal{C}. \tag{3.54}$$

Constraints (3.54) state that in a feasible integer solution, the number of paths that perform two or more tasks in a subset $\mathcal{S} \subseteq \mathcal{C}$ must be less than or equal to 1. Adding these inequalities to the master problem changes the formulation of the pricing problem. Let $\mathcal{S}$ be the subset of triplets of customers for which the subset row inequality has been generated and added to the master problem. Also, let $\beta_{\mathcal{S}} \leq 0$ be the dual variable associated with subset $\mathcal{S}$. Then, the pricing problem is formulated as:

$$\min_{p \in \mathcal{P}} \left\{ \mathfrak{r}_p = c_p - \sum_{i \in \mathcal{C}} a_{ip} \pi_i - \sum_{o \in \mathcal{O}} b_{op} \sigma_o - \sum_{\mathcal{S} \in \mathcal{S}} \beta_{\mathcal{S}} \left\lfloor \frac{1}{2} \sum_{i \in \mathcal{S}} a_{ip} \right\rfloor \right\}. \tag{3.55}$$

This formulation is usually more challenging to solve, particularly if the number of subset row inequalities already added to the master problem is high. As a consequence, the cut separator follows three steps. First, it enumerates all task triplets and checks if the current fractional solution violates the associated inequality by at least ε units. If so, the inequality is added to a list. After this step is completed, the list is sorted ensuring that inequalities with greater violations remain on top. Finally, the cut separator adds the first φ inequalities to the master problem. We set φ to 10 and ε to 0.1.

The PA used to solve the pricing problem handles these inequalities by adding a new resource for each subset $\mathcal{S} \in \mathcal{S}$. These resources keep track of the number of tasks in the subset that have been fulfilled. Every time that one of these resources reaches a value of 2, the associated dual variable $\beta_{\mathcal{S}}$ is subtracted from the objective function.

3.4.5 Branching strategy

Even after adding inequalities, the optimal solution of the master problem can remain fractional. In such a case, the algorithm uses a set of branching rules. We implement a five-stage hierarchical branching. At all levels, we branch on the variable for which the fractional value is closer to 0.5. The first level branches on driving flow variables. We branch on the implicit variable x_{ij} which equals 1 if any team drives through arc $(i, j) \in \mathcal{A}$, and 0 otherwise. To enforce $x_{ij} = 0$ we remove all the paths that involve driving between tasks i and j from the master problem and we also forbid the PA to use the arc while solving the pricing problem. To enforce $x_{ij} = 1$ we remove all the paths in the master problem that fulfill tasks i or j without using arc (i, j). In addition, we forbid the PA to use any arc starting at node i and ending at any node different than j. We also forbid the PA to use any arc ending at node j and starting at any node different than i. The second level branches on walking flow variables. Similarly, we branch on the implicit variable h_{ij} that takes the value of 1 if any team walks through arc $(i, j) \in \mathcal{A}$, and 0 otherwise. Decisions are enforced in the same manner as for the first level.

The third level branches on the assignment of tasks to workers. More specifically, we branch on the implicit variables y_i^k which take 1 if worker $k \in \mathcal{W}$ fulfills task $i \in \mathcal{C}$. To enforce $y_i^k = 0$ we remove all the paths that fulfill task i and include worker k in the team. Also, we forbid the PA to fulfill task i if the worker is currently in the team. To enforce $y_i^k = 1$ we remove all the paths that fulfill task i without including worker k in the team. In addition, we remove the possibility of outsourcing task i and we forbid the PA to fulfill task i using teams that do not include worker k. The fourth level branches on the selection of workers. More specifically, we branch on the implicit variables v_k which equals 1 if worker $k \in \mathcal{W}$ is included in any path of the current solution, and 0 otherwise. To guarantee that $v_k = 0$, we remove all the paths that include worker k in the team and remove the corresponding node from the modified network. To enforce $v_k = 1$, we add a constraint to the master

problem. The fifth level branches on the ϑ_i variables to either forbid or enforce the outsourcing of task $i \in C$. To enforce this branching rule, we add a constraint to the master problem. The algorithm explores the enumeration branch-and-bound tree using a best-bound search strategy.

3.4.6 General enhancements

In the following sections, we present key strategies that we use to enhance the performance of our BPC algorithm.

Heuristic and leveled pricing

Solving the pricing problem (3.42) to optimality can be a challenging task. Fortunately, the pricing problem does not have to be solved to optimality at every iteration (Parragh and Cordeau 2017). Instead, it is sufficient to find at least one path with negative reduced cost. Following this intuition, we allow the PA to heuristically terminate the search if one of the following two conditions holds:

- The PA has found at least Υ paths with negative reduced cost;
- The PA has found at least one path with negative reduced cost and the computational time spent during the search is higher than Λ.

In a first set of experiments with our BPC algorithm, we observed that some paths inside the pool were almost identical. More precisely, they included the same subset of tasks but were using different team of workers. These paths were usually generated in different iterations of the pricing problem. This may happen for two reasons. First, it may take a while before the PA backtracks completely from the end depot to the source node to select a different subset of workers. Second, because we allow the PA to heuristically terminate the search, thus not all paths will be necessarily considered before the PA stops. To address this issue, every time we find a path with negative reduced cost, we check if there are other teams of workers

that could perform the same subset of tasks and add them to the pool. In this step, we only consider the teams of workers that are currently used in the solution of the master problem. This strategy saves time on the pricing problem and also has the potential to decrease the number of column generation iterations.

To further speed up the solution of the pricing problem, we use a leveled pricing strategy. This strategy consists of solving the pricing problem in two stages. In the first stage, we solve the pricing problem without considering the possibility of walking between two locations. This means that lines 10 to 14 of the Algorithm 8 are temporarily ignored. As a result, during this stage, walking subtours are not allowed. Only if the PA was not able to find paths with negative reduced cost during the first stage, we proceed to the following stage. In the second stage, we solve the pricing problem while considering both driving and walking. In addition, following the ideas of Lozano and Medaglia (2013), our PA propagates different partial paths in parallel. In practice, we trigger multiple threads at the source node $\mathfrak{s}$ that begin to propagate partial paths starting from different outgoing arcs. This allows the algorithm to test different team configurations almost simultaneously. Similarly, once a partial path reaches the start depot $\overline{0}$, it also triggers multiple threads that propagate partial paths using the current team configuration.

Initial set of paths

The master problem is always feasible as it is always possible to outsource all tasks. Thus, it is possible to start the BPC algorithm with an empty set of paths. Several researchers have observed that the performance of BPC algorithms can be affected by the initial set of variables. In particular, building an initial solution can help to better estimate the values of the dual variables associated with each constraint of the master problem, which may reduce the possibility of producing irrelevant columns during early iterations (Lübbecke and Desrosiers 2005). Thus, it is a common practice to use fast heuristics to warm-start BPC algorithms. In our algorithm, we use an adaptation of the constructive heuristic proposed by Xie et al. (2017) to quickly

generate paths carried out by single-worker teams.

Algorithm 10 presents the main logic of the `build_initial_solution` function. From lines 1 to 8, the algorithm iterates over the set of tasks $\mathcal{C}$. Line 2 computes the angle $\tilde{\theta}_i$ between the geographical location of each task $i \in \mathcal{C}$ and a horizontal line drawn from the depot's location. From lines 3 to 7, the algorithm iterates over the set of workers $\mathcal{W}$. Line 4 checks if task i can be performed by worker k. If so, the number of tasks $\tilde{\sigma}_k$ that worker k can perform is increased by one. Then, line 9 constructs list $\mathscr{R}_1$ by sorting the workers in $\mathcal{W}$ in non-increasing order of the number of tasks they are qualified to perform $\tilde{\sigma}_k$. Similarly, line 10 constructs list $\mathscr{R}_2$ by sorting every task $i \in \mathcal{C}$ in non-decreasing order of the angle $\tilde{\theta}_i$. After constructing both lists, from lines 11 to 19 the algorithm iterates over each task in list $\mathscr{R}_2$. Line 12 initially sets task i as outsourced (i.e., $\vartheta_i = 1$). Then, from lines 13 to 18, the algorithm iterates over each worker in list $\mathscr{R}_1$, as long as the task is still set as outsourced. Line 14 evaluates the feasibility of inserting the task into the path of the current worker. If the insertion is feasible, line 15 sets the task as completed (i.e., $\vartheta_i = 0$), and line 16 inserts the task into the path p_k of worker k. Line 20 returns the initial set of paths.

Algorithm 10 `build_initial_solution` function

Require: $\mathcal{C}$, Set of tasks; $\mathcal{W}$, Set of workers; $\overline{0}$, Depot.
Ensure: $\mathscr{P}^*$, optimal path.

for $i \in \mathcal{C}$ **do**
  $\tilde{\theta}_i \leftarrow$ `compute_angle`$(i, \overline{0})$
  for $k \in \mathcal{W}$ **do**
    if `evaluate_skills`(i, k) **then**
      $\tilde{\sigma}_k \leftarrow \tilde{\sigma}_k + 1$
    end if
  end for
end for
$\mathscr{R}_1 \leftarrow$ `sort_non_increasing`$(\mathcal{W}, \tilde{\sigma})$
$\mathscr{R}_2 \leftarrow$ `sort_non_decreasing`$(\mathcal{C}, \tilde{\theta})$
for $i \in \mathscr{R}_2$ **do**
  $\vartheta_i \leftarrow 1$
  for $k \in \mathscr{R}_1 | \vartheta_i = 1$ **do**
    if `evaluate_insertion`(i, p_k) **then**
      $\vartheta_i \leftarrow 0$
      $p_k \leftarrow$ `insert`(i, k)
    end if
  end for
end for
return $\bigcup_{k \in \mathcal{W}} p_k$

3.5 Computational experiments

The proposed BPC algorithm was implemented in Java using the jORLib[1] library and compiled using Java 1.8.0 331. We rely on CPLEX 20.1 to solve the master problem. All the experiments were conducted on the Beluga cluster of the Digital Research Alliance of Canada using eight threads and 20GB of RAM in a Linux environment. The time limit for all the experiments is 2 hours. After fine tuning, we set the bound step size in the PA to 10. The bounding time limits are set to 0.2ϕ and ϕ. The maximum number of paths Υ is set to 10 and the time limit Λ to 5 seconds. All the instances and solutions are available at `https://chairelogistique.hec.ca/en/scientific-data/`.

[1]The latest version of jORLib can be downloaded at: http://coin-or.github.io/jorlib/.

3.5.1 Set of instances

We created a set of instances based on the testbed designed by Kovacs et al. (2012) for the STRSP. These instances were derived from the well-known VRPTW instances proposed by Solomon (1987). There are three classes of instances (`D`), denoted as random (R), clustered (C), and semi-clustered (RC). For each class of instances, two types of planning horizons (`T`) were considered, namely, short (1) and long (2). In addition, the percentage of tasks that have a time window constraint (`m`) can take two values, where 01 is used to indicate 100% and 03 to specify 50%. To fulfill tasks, two sets of workers (`E`) are considered: complete (C) and reduced (R). These sets differ in the number of workers available. In the complete set, up to 130 workers are available. In the reduced set, up to 25. The number of skills domains (`S`) is selected in the set $\{5, 6, 7\}$ and each skill may have different levels of proficiency (`L`) selected from the set $\{4, 6\}$. Finally, the number of tasks (`n`) is selected in the set $\{25, 50, 75\}$. A total of 162 instances are considered. An instance is referred to as "`DTm_E_S×L_n`".

For each instance, the skills requirements ν_{iql}, the time window $[a_i, b_i]$, the outsourcing cost f_i, and service time associated with each task $i \in \mathcal{C}$ are known. Similarly, information regarding the qualifications ξ_{kql} of each worker $k \in \mathcal{W}$ is given. The driving distance (in kilometers) μ_{ij} between nodes (tasks and depot) is computed using the Euclidean distance. The maximum duration of each route corresponds to the latest arrival time to the depot $b_{\underline{0}}$. To extend this set of instances to the WSRP-PL, we compute driving and walking times considering a driving speed of 60 km/h and a walking speed of 4 km/h. We assume that driving and walking distances are equal. The maximum number of workers per team can be either two or three and we fixed $|\mathcal{T}| = m$. In addition, we consider a maximum daily walking distance ζ for each worker of 5 km. The maximum walking distance between two nodes θ is set to 2.5 km. Finally, the variable cost c^v is set to 1.

3.5.2 Experiment 1: assessing the BPC performance

To assess the effectiveness and efficiency of our BPC algorithm, we present a comparison with the solution of the arc-based formulation described in Section 3.2, which is the only other exact method available to solve the WSRP-PL. This formulation was solved using CPLEX 20.1. In preliminary experiments, we used CPLEX indicator constraints to express the big-M constraints (3.16)-(3.18). However, this did not lead to an improvement in the performance of AF.

Tables 3.1 and 3.2 present the detailed results of the comparison between the BPC algorithm and the arc-based formulation (AF) setting the maximum number of workers in a team to 2 and 3, respectively. Column 1 denotes the set of workers. Column 2 provides information regarding the instance class and the percentage of tasks having a time window. Column 3 gives the number of tasks. The remaining columns give the number of optimal solutions found, the average optimality gap, and the computational time in seconds used by each algorithm. To compute the optimality gap we use the best lower bound known for each instance (i.e., the highest lower bound between the bounds found by AF and BPC).

Tables 3.1 and 3.2 present the detailed results of the comparison between the BPC algorithm and the arc-based formulation (AF) setting the maximum number of workers in a team to 2 and 3, respectively. Column 1 denotes the set of workers. Column 2 provides information regarding the instance class and the percentage of tasks having a time window. Column 3 gives the number of tasks. The remaining columns give the number of optimal solutions found, the average optimality gap, and the computational time in seconds used by each algorithm. To compute the optimality gap we use the best lower bound known for each instance (i.e., the highest lower bound between the bounds found by AF and BPC). Whenever AF ran out of memory, we used the last integer solution found to compute the optimality gap.

Table 3.1 shows that BPC can solve 138 out of 162 instances to optimality, while AF can only solve 79. Remarkably, BPC can solve all the instances with 25 tasks

Table 3.1: Comparison between AF and BPC on the WSRP-PL instances with $\omega = 2$.

E	Tm	n	AF			BPC		
			#Opt.	Avg. Δ	CPU (s)	#Opt.	Avg. Δ	CPU (s)
C	101	25	8/9	0.31%	1119.66	9/9	0.00%	293.41
C	103	25	3/9	14.68%	7200.00	9/9	0.00%	306.25
C	201	25	7/9	2.10%	2943.23	9/9	0.00%	220.25
C	101	50	6/9	9.21%	4168.58	8/9	0.44%	1145.64
C	103	50	0/9	33.07%	7200.00	7/9	9.09%	2389.94
C	201	50	4/9	14.32%	5420.10	9/9	0.00%	687.24
C	101	75	4/9	22.09%	5631.28	9/9	0.00%	1464.49
C	103	75	0/9	46.37%	7200.00	5/9	12.50%	3677.19
C	201	75	1/9	31.94%	6413.09	5/9	15.54%	3760.98
R	101	25	9/9	0.00%	844.16	9/9	0.00%	67.01
R	103	25	6/9	5.45%	7200.00	9/9	0.00%	590.97
R	201	25	9/9	0.00%	650.00	9/9	0.00%	88.68
R	101	50	7/9	0.42%	3064.41	9/9	0.00%	360.22
R	103	50	1/9	14.47%	7200.00	6/9	2.53%	2466.72
R	201	50	7/9	4.46%	1953.28	7/9	6.21%	1632.36
R	101	75	4/9	9.37%	4159.29	7/9	0.21%	1893.61
R	103	75	0/9	30.26%	7200.00	5/9	12.45%	4055.22
R	201	75	3/9	8.01%	5648.70	7/9	22.22%	2345.15
Total/Avg.			79/162	13.70%	4734.21	138/162	4.51%	1524.74

to optimality. As expected, instances in which the percentage of tasks having a time window is lower (i.e., Tm = 103) are harder to solve. This is especially the case for AF, as it can only solve 10 out of 54 instances with this setting. A similar argument can be used for the subset of instances with wider time windows (i.e., Tm = 201). As routes are longer, BPC, a column generation-based method, decreases its performance. With regard to the number of workers available, it seems that AF better handles instances with a lower number of workers (i.e., E = R), as it can solve 12 more instances (46 vs 33) compared with the subset of instances with the complete set of workers. One plausible explanation is that the reduction in the number of variables has a strong positive impact on AF's performance. With respect to the computational times, on average BPC takes 1524.74 seconds to solve the WSRP-PL, while AF takes 4734.21 seconds. BPC is particularly fast on the

subset of instances with 25 customers.

Table 3.2: Comparison between AF and BPC on the WSRP-PL instances with $\omega = 3$.

E	Tm	n	AF			BPC		
			#Opt.	Avg. Δ	CPU (s)	#Opt.	Avg. Δ	CPU (s)
C	101	25	8/9	2.21%	1155.05	8/9	1.83%	2482.04
C	103	25	1/9	23.74%	7200.00	5/9	15.90%	4415.58
C	201	25	7/9	3.02%	4135.11	9/9	0.00%	3295.42
C	101	50	4/9	11.27%	4983.59	6/9	1.58%	4417.96
C	103	50	0/9	51.22%	7200.00	3/9	19.23%	5795.05
C	201	50	1/9	16.17%	5733.85	6/9	6.85%	3810.50
C	101	75	2/9	21.66%	6513.46	7/9	12.28%	4437.02
C	103	75	0/9	70.86%	7200.00	2/9	45.78%	5951.84
C	201	75	1/9	34.23%	6415.43	2/9	15.45%	6401.91
R	101	25	9/9	0.00%	462.03	9/9	0.00%	229.99
R	103	25	4/9	4.22%	7200.00	9/9	0.00%	360.98
R	201	25	9/9	0.00%	432.66	9/9	0.00%	410.75
R	101	50	5/9	1.90%	4274.53	9/9	0.00%	1682.59
R	103	50	0/9	15.40%	7200.00	6/9	1.57%	3730.21
R	201	50	4/9	4.35%	4176.98	6/9	3.87%	3030.08
R	101	75	2/9	16.54%	6414.02	3/9	4.00%	5222.80
R	103	75	0/9	50.31%	7200.00	2/9	29.92%	6021.39
R	201	75	4/9	12.07%	5813.69	2/9	14.31%	6347.66
Total/Avg.			61/162	18.84%	5206.13	103/162	9.59%	3780.15

Table 3.2 shows that BPC can solve 103 out of 162 instances to optimality, while AF only solves 61. The average optimality gap of the solutions retrieved by BPC is 9.59% while the average optimality gap of the solutions found by AF is 18.84%. With respect to the computational times, on average BPC takes 3780.15 seconds to solve the WSRP-PL, while AF takes 5206.13 seconds. Note, that increasing the maximum number of workers per team increases the difficulty of the problem. This is expected because the possible number of teams that can be formed is significantly larger.

3.5.3 Experiment 2: analyzing the BPC components

In order to evaluate the contribution of some of the components of our algorithm, we compare three variants of our BPC algorithm. The first variant, labeled as BAP (i.e., branch and price), corresponds to a version of the BPC algorithm that does not use the cut separator. The second variant, labeled as BPC-WI, is a version of the BPC algorithm without the initialization procedure described in Section 3.4.6. Finally, the third variant, labeled as BPC, corresponds to the full version of our branch-price-and-cut algorithm.

Table 3.3 compares the performance of each variant. Each row in the table corresponds to an algorithm. Columns 2, 3, and 4 contain the average optimality gap, the average number of cuts, and the average number of columns generated at the root node. Column 5 reports the average number of branch-and-price nodes solved. Column 6 shows the average number of cuts generated. Column 7 gives the average number of column generation iterations. Column 8 reports the average number of columns generated during the execution of the algorithm. Columns 9 and 10 show the average computational time spent in the master problem and the average total computational time in seconds. Finally, column 11 reports the number of instances solved to optimality.

Table 3.3: Comparison of the BPC variants.

Variant	Root node			Nodes	Cuts	Iters	Cols	Master	Total	# Opt.
	Gap (%)	Cuts	Cols					CPU (s)	CPU (s)	
BAP	15.3	0.0	1461.7	23.9	0.0	448.1	3977.5	2.6	3008.1	221/324
BPC-WI	14.0	5.4	1615.9	3.1	10.3	208.7	2719.3	2.1	2835.0	232/324
BPC	12.3	5.6	1530.9	2.9	10.5	202.1	2641.7	1.9	2609.9	241/324

As the results show, the BAP variant of the algorithm usually explores a larger branch-and-price tree. In particular, on average BAP solves almost 7 times more branch-and-price nodes than the variants of the algorithm that use a cut separator. As a consequence, the BAP algorithm requires almost twice as many column generation iterations to solve an instance. Furthermore, the number of columns (i.e.,

paths) generated is at least 1.5 times as large as the ones generated by the other two variants. With respect to the impact of the initialization procedure, the average optimality gap at the root node decreases by 1.7%. This can be explained by the fact that BPC has early access to a stronger upper bound. The impact of the initialization procedure is also shown on the average number of columns generated both at the root node and in the complete branch-and-price tree. On average, BPC is the algorithm that generates the fewest columns. In terms of the computational times, the three variants of the algorithm have a similar performance. However, BPC solves 20 and 9 more instances to optimality than BAP and BPC-WI, respectively.

3.5.4 Experiment 3: analyzing the computational impact of park-and-looping

We set up an experiment to shed some light into the computational impact of allowing park-and-loop routes. To accomplish this goal we ran our BPC on the whole set of instances setting $\zeta = 0$ (i.e., forbidding the walking subtours). Next, we compared the results to those obtained when park-and-looping is allowed (see Experiment 1). Table 3.4 summarizes the results delivered by BPC running with (BPC) and without (BPC-D) park-and-looping. Each row corresponds to a combination of a set of workers, the maximum number of workers per team, and the number of tasks. Columns 4 and 7 report the number of optimal solutions found in each setting. Columns 5 and 8 show the average optimality gap. Columns 6 and 9 contain the average CPU time in seconds.

The results indicate that allowing park-and-looping significantly increases the difficulty of the problem. Out of the 324 instances, BPC-D proves optimality on 267, while BPC achieves it in 241. Moreover, BPC-D is able to establish optimality in all 25-task instances and in 95 out of 108 of the 50-task instances. A closer examination of the optimality gaps and CPU times confirms the conclusion. While BPC-D reports an average gap of 3.35%, BPC reports a nearly twofold average gap

Table 3.4: Comparison of BPC performance on the WSRP-PL with and without allowing park-and-loop routes.

E	Tm	n	BPC-D			BPC		
			#Opt.	Avg. Δ	CPU (s)	#Opt.	Avg. Δ	CPU (s)
C	101	25	18/18	0.00%	773.75	17/18	0.91%	1387.73
C	103	25	17/18	0.14%	733.82	14/18	7.95%	2360.92
C	201	25	18/18	0.00%	835.73	18/18	0.00%	1757.84
C	101	50	16/18	1.19%	2036.43	14/18	1.01%	2781.80
C	103	50	13/18	5.90%	3038.96	10/18	14.16%	4093.49
C	201	50	17/18	0.42%	1730.15	15/18	3.43%	2284.87
C	101	75	17/18	0.88%	2359.70	16/18	6.14%	2950.76
C	103	75	9/18	12.83%	4425.10	7/18	29.14%	4814.51
C	201	75	10/18	4.56%	4791.78	7/18	15.49%	5081.45
R	101	25	18/18	0.00%	91.39	18/18	0.00%	147.00
R	103	25	18/18	0.00%	270.48	18/18	0.00%	475.98
R	201	25	18/18	0.00%	60.89	18/18	0.00%	249.72
R	101	50	18/18	0.00%	850.00	18/18	0.00%	1021.41
R	103	50	15/18	1.15%	1908.46	12/18	2.05%	3098.46
R	201	50	14/18	3.27%	2108.47	13/18	5.04%	2331.22
R	101	75	11/18	2.27%	3822.91	10/18	2.10%	3558.20
R	103	75	8/18	19.27%	4624.48	7/18	21.19%	5038.30
R	201	75	12/18	8.35%	3335.90	9/18	18.27%	4346.41
Total/Avg.			267/324	3.35%	2099.91	241/324	7.05%	2652.45

of 7.05%. The former also runs (on average) nearly 30% faster (2099.91 vs. 2652.45 seconds).

3.5.5 Experiment 4: measuring the impact of the subtour transportation mode

In this experiment, we aim to measure the impact of using a faster and longer-range transportation mode (e.g., an electric bike or scooter) to perform the subtours. More specifically, we compare the solutions we obtained in Experiment 1 with those obtained for each instance while increasing the walking speed (10 km/h) and the maximum daily walking distance (20 km) to mimic the behavior of an electric scooter.

Table 3.5 compares the solutions found by BPC for each instance when consid-

ering both configurations. We denote the configuration that mimics the e-scooter behavior as BPC-S. The first three columns provide information regarding the set of workers, the instance series, and the number of customers. Columns 4 and 9 report the number of optimal solutions found by BPC under each configuration. Columns 5 and 10 present the average optimality gap. The optimality gap is computed using the lower bound found by each algorithm. Columns 6 and 11 report the average computational time in seconds. Columns 7 and 12 present the average number of subtours in each solution. Columns 8 and 13 report the average maximum walking distance. Finally, Column 14 presents the average gap in the objective function when both BPC and BPC-S solved the instance to optimality. More specifically, for each instance in the set, the gap was computed as

$$\frac{f(BPC\text{-}S) - f(BPC)}{f(BPC)}, \tag{3.56}$$

where $f(\cdot)$ is the objective function found by BPC under each configuration.

Table 3.5: Comparison of BPC performance while varying the walking speed and the maximum walking distance.

E	Tm	n	BPC					BPC-S					Δ OF
			#Opt.	Avg. Δ	CPU (s)	#S.	WD	#Opt.	Avg. Δ	CPU (s)	#S.	WD	
C	101	25	17/18	0.91%	1387.73	0.72	2.39	13/18	3.49%	2511.84	1.72	12.14	-0.49%
C	103	25	14/18	7.95%	2360.92	1.06	2.25	13/18	16.50%	2783.15	2.11	11.78	-0.72%
C	201	25	18/18	0.0%	1757.84	0.61	1.33	14/18	7.11%	2929.02	3.39	17.20	-2.66%
C	101	50	14/18	1.01%	2781.80	0.56	1.16	10/18	6.93%	4201.41	3.22	13.22	-0.54%
C	103	50	10/18	14.16%	4093.49	0.78	1.14	8/18	23.15%	4741.30	2.56	12.42	-0.69%
C	201	50	15/18	3.43%	2248.87	0.67	1.56	7/18	21.01%	4919.81	3.83	17.69	-1.83%
C	101	75	16/18	6.14%	2950.75	0.67	1.19	8/18	12.05%	4807.59	4.17	15.90	-0.85%
C	103	75	7/18	29.14%	4814.51	0.41	0.50	5/18	30.70%	5913.82	2.33	10.94	-0.19%
C	201	75	7/18	15.49%	5081.45	0.44	1.11	4/18	35.26%	6217.24	2.56	12.36	-0.59%
R	101	25	18/18	0.00%	147.00	0.78	2.61	17/18	0.03%	805.21	2.50	13.48	-1.12%
R	103	25	18/18	0.00%	475.98	1.11	2.27	17/18	4.91%	1087.30	2.61	12.57	-2.75%
R	201	25	18/18	0.00%	249.72	0.83	2.00	18/18	0.00%	350.98	3.83	19.35	-3.35%
R	101	50	18/18	0.00%	1021.41	0.72	1.39	15/18	1.07%	1955.17	3.67	14.94	-0.58%
R	103	50	12/18	2.05%	3098.46	1.11	1.80	12/18	12.67%	3490.83	3.89	16.69	-0.90%
R	201	50	13/18	5.04%	2331.22	0.72	1.78	10/18	15.89%	3713.64	4.39	17.86	-0.61%
R	101	75	10/18	2.10%	3558.20	0.78	1.39	8/18	5.72%	4192.80	4.22	12.58	-0.42%
R	103	75	7/18	21.19%	5038.30	0.67	0.72	5/18	22.83%	5491.86	2.28	9.91	-0.34%
R	201	75	9/18	18.27%	4346.41	0.69	1.25	5/18	24.19%	5373.63	3.17	15.73	-0.44%
Total/Avg.			241/324	7.05%	2652.45	0.74	1.55	189/324	13.53%	3638.14	3.14	14.26	-1.06%

As the results show, increasing the speed and the range of the mode employed for the subtours has a positive impact on the objective function. On average, the

objective function decreases by 1.06%, but the savings are probably greater. Note that BPC-S only solves to optimality 189 out of 324 instances while BPC solves 241 out of 324. As a result, the average optimality gap reported by BPC-S is 13.53%, while BPC solutions have an average optimality gap of 7.05%. These figures also indicate that solving an instance of the WSRP-PL under the second configuration is significantly harder. A possible explanation for this behavior is given by the average number of subtours performed in each route under both settings. Indeed, while BPC finds solutions with an average of 0.74 subtours, BPC-S builds solutions with 3.14 subtours on average.

3.5.6 Experiment 5: solving the no-team STRSP

In Experiment 1, we demonstrated that our BPC can optimally solve WSRP-PL instances with up to 75 nodes. The most closely related problem, featuring publicly available instances of comparable size, is the no-team version of the STRSP. Note that our WSRP-PL reduces to that problem when $\omega = 1$ and $\zeta = 0$. We, therefore, ran our AF and BPC on the set of large 100-task instances proposed by Kovacs et al. (2012), available at `http://prolog.univie.ac.at/research/STRSP/` and compared our results to the state of the art.

Table 3.6 and 3.7 compare the performance of BPC and AF on the no-team STRSP instances when the set of workers is complete and reduced, respectively. Columns 1 and 2 denote the class of the instance and the instance series. Column 3 presents the average number of workers available. Columns 4 and 7 report the number of optimal solutions found by each algorithm. Columns 5 and 8 show the average optimality gap. Columns 6 and 9 contain the average CPU time employed by each algorithm in seconds.

Tables 3.6 and 3.7 clearly indicate that in this problem our BPC also outperforms AF. In the subset of instances in which the complete set of workers is available, BPC solves 13 out of 27 instances to optimality, while AF solves none. The solution found

Table 3.6: Comparison between AF and BPC on the Kovacs et al. (2012) 100-task complete instances of the no-team STRSP.

D	Tm	$\lvert\mathcal{W}\rvert$	AF			BPC		
			#Opt.	Avg. Δ	CPU (s)	#Opt.	Avg. Δ	CPU (s)
C	101	16.7	0/3	24.68%	7200.0	3/3	0.00%	929.9
R	101	26.3	0/3	7.19%	7200.0	3/3	0.00%	717.9
RC	101	23	0/3	68.52%	7200.0	3/3	0.00%	1027.5
C	103	16.7	0/3	91.76%	7200.0	0/3	4.74%	7200.0
R	103	26.3	0/3	96.28%	7200.0	1/3	0.36%	4929.0
RC	103	23	0/3	95.01%	7200.0	0/3	4.97%	7200.0
C	201	7.67	0/3	9.22%	7200.0	2/3	5.29%	2425.5
R	201	8	0/3	35.78%	7200.0	1/3	17.79%	4855.9
RC	201	8.67	0/3	55.01%	7200.0	0/3	6.37%	7200.0
	Total/Avg.		0/27	53.72%	7200.0	13/27	4.39%	4054.0

Table 3.7: Comparison between AF and BPC on the Kovacs et al. (2012) 100-task reduced instances of the no-team STRSP.

D	Tm	$\lvert\mathcal{W}\rvert$	AF			BPC		
			#Opt.	Avg. Δ	CPU (s)	#Opt.	Avg. Δ	CPU (s)
C	101	4.67	0/3	33.42%	7200.0	3/3	0.00%	102.7
R	101	4.67	0/3	34.15%	7200.0	3/3	0.00%	398.1
RC	101	4.67	0/3	77.86%	7200.0	3/3	0.00%	3514.4
C	103	4.67	0/3	92.90%	7200.0	1/3	14.84%	6750.8
R	103	4.67	0/3	93.83%	7200.0	1/3	3.62%	4835.2
RC	103	4.67	0/3	91.70%	7200.0	0/3	16.52%	7200.0
C	201	4.67	2/3	2.23%	2583.3	0/3	18.38%	7200.0
R	201	4.67	0/3	39.15%	7200.0	0/3	40.75%	7200.0
RC	201	4.67	0/3	50.25%	7200.0	0/3	29.66%	7200.0
	Total/Avg.		2/27	57.28%	6683.1	11/27	13.75%	4933.5

by BPC improves the best known solution reported by Gu et al. (2022) in 3 of these 13 instances. Moreover, the average optimality gap of the solutions found by BPC is 4.39% while AF reports solutions with an average optimality gap of 53.72%. With respect to the computational times, AF always reached the time limit. In contrast, BPC uses on average 4054 seconds. A similar situation is observed in the subset of instances with a reduced set of workers. AF solves 2 out of 27 instances to optimality, while BPC solves 11. The solution found by BPC is the new best known solution for 9 of these instances. On average the optimality gap of the solutions provided by

AF is 57.28% while BPC finds solutions with an average optimality gap of 13.75%. The objective function of the new best known solutions found by BPC is available in Appendix C.

3.6 Concluding remarks

In this paper, we introduced the workforce scheduling and routing problem with park-and-loop. Routing decisions can be particularly challenging as routes can include one or more subtours that are covered on foot. To solve this problem we presented a branch-price-and-cut algorithm that is capable of solving instances with up to 75 tasks and 130 workers in less than two hours. The algorithm also provided optimal solutions for the closely related technician routing and scheduling problem.

Extensive computational experiments carried out on instances derived from a popular benchmark in the literature suggest that the proposed algorithm can outperform a standard arc-based formulation both in terms of solution quality and running times. Indeed, our algorithm solved 241 out of 324 instances to optimality while the arc-based formulation only solved 140. This performance gap comes as a result of using state-of-the-art techniques and exploiting problem-specific features to enhance the pricing problem algorithm.

To encourage further research on the WSRP-PL we designed an online tool where all the instances and solutions are available. Further research will focus on a more challenging variant of the problem in which teams can partially split (i.e., workers can fulfill a task on their own and then rejoin the team). In some cases, splitting a team may be beneficial to the company's efficiency. However, it can easily disrupt the structure of the routes, that do not longer follow a regular park-and-loop structure.

References

Braekers, K., R. F. Hartl, S. N. Parragh, and F. Tricoire (2016). "A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience". *European Journal of Operational Research* 248.2, pp. 428–443.

Bredström, D. and M. Rönnqvist (2008). "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints". *European Journal of Operational Research* 191.1, pp. 19–31.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2022). "The doubly open park-and-loop routing problem". *Computers & Operations Research* 143, p. 105761.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2023). "Solving the park-and-loop routing problem by branch-price-and-cut". *Transportation Research Part C: Emerging Technologies* 157, p. 104369.

Cabrera, N., A. L. Medaglia, L. Lozano, and D. Duque (2020). "An exact bidirectional pulse algorithm for the constrained shortest path". *Networks* 76.2, pp. 128–146.

Castillo-Salazar, J. A., D. Landa-Silva, and R. Qu (2016). "Workforce scheduling and routing problems: literature survey and computational study". *Annals of Operations Research* 239, pp. 39–67.

Chen, X., B. W. Thomas, and M. Hewitt (2016). "The technician routing problem with experience-based service times". *Omega* 61, pp. 49–61.

Coindreau, M.-A., O. Gallay, and N. Zufferey (2019). "Vehicle routing with transportable resources: Using carpooling and walking for on-site services". *European Journal of Operational Research* 279, pp. 996–1010.

Contardo, C. and R. Martinelli (2014). "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". *Discrete Optimization* 12, pp. 129–146.

Costa, L., C. Contardo, and G. Desaulniers (2019b). "Exact branch-price-and-cut algorithms for vehicle routing". *Transportation Science* 53.4, pp. 946–985.

Dellaert, N., F. Dashty Saridarq, T. Van Woensel, and T. G. Crainic (2019). "Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows". *Transportation Science* 53.2, pp. 463–479.

Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen (2004). "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". *Networks* 44.3, pp. 216–229.

Goel, A. and F. Meisel (2013). "Workforce routing and scheduling for electricity network maintenance with downtime minimization". *European Journal of Operational Research* 231.1, pp. 210–228.

Gu, H., Y. Zhang, and Y. Zinder (2022). "An efficient optimisation procedure for the workforce scheduling and routing problem: Lagrangian relaxation and iterated local search". *Computers & Operations Research* 144, p. 105829.

Guastaroba, G., J.-F. Côté, and L. C. Coelho (2021). "The multi-period workforce scheduling and routing problem". *Omega* 102, p. 102302.

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

Kovacs, A. A., S. N. Parragh, K. F. Doerner, and R. F. Hartl (2012). "Adaptive large neighborhood search for service technician routing and scheduling problems". *Journal of Scheduling* 15, pp. 579–600.

Le Colleter, T., D. Dumez, F. Lehuédé, and O. Péton (2023). "Small and large neighborhood search for the park-and-loop routing problem with parking selection". *European Journal of Operational Research* 308.3, pp. 1233–1248.

Lozano, L. and A. L. Medaglia (2013). "On an exact method for the constrained shortest path problem". *Computers & Operations Research* 40.1, pp. 378–384.

Lübbecke, M. E. and J. Desrosiers (2005). "Selected topics in column generation". *Operations Research* 53.6, pp. 1007–1023.

Mendoza, J. E. and J. G. Villegas (2013). "A multi-space sampling heuristic for the vehicle routing problem with stochastic demands". *Optimization Letters* 7, pp. 1503–1516.

Paraskevopoulos, D. C., G. Laporte, P. P. Repoussis, and C. D. Tarantilis (2017). "Resource constrained routing and scheduling: Review and research prospects". *European Journal of Operational Research* 263.3, pp. 737–754.

Parragh, S. N. and J.-F. Cordeau (2017). "Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows". *Computers and Operations Research* 83, pp. 28–44.

Pereira, D. L., J. C. Alves, and M. C. de Oliveira Moreira (2020). "A multiperiod workforce scheduling and routing problem with dependent tasks". *Computers & Operations Research* 118, p. 104930.

Pessoa, A., R. Sadykov, and E. Uchoa (2018). "Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems". *European Journal of Operational Research* 270.2, pp. 530–543.

Schrotenboer, A. H., E. Ursavas, and I. F. Vis (2019). "A branch-and-price-and-cut algorithm for resource-constrained pickup and delivery problems". *Transportation Science* 53.4, pp. 1001–1022.

Solomon, M. M. (1987). "Algorithms for the vehicle routing and scheduling problems with time window constraints". *Operations Research* 35.2, pp. 254–265.

Tricoire, F., N. Bostel, P. Dejax, and P. Guez (2013). "Exact and hybrid methods for the multiperiod field service routing problem". *Central European Journal of Operations Research* 21, pp. 359–377.

Xie, F., C. N. Potts, and T. Bektaş (2017). "Iterated local search for workforce scheduling and routing problems". *Journal of Heuristics* 23, pp. 471–500.

Zamorano, E., A. Becker, and R. Stolletz (2018). "Task assignment with start time-dependent processing times for personnel at check-in counters". *Journal of Scheduling* 21, pp. 93–109.

Zhou, H., H. Qin, C. Cheng, and L.-M. Rousseau (2023). "An exact algorithm for the two-echelon vehicle routing problem with drones". *Transportation Research Part B: Methodological* 168, pp. 124–150.

Chapter 4

The Dynamic Park-and-loop Routing Problem

Abstract

This paper introduces the dynamic park-and-loop routing problem, an important practical extension of the park-and-loop routing problem in which some of the customer demand is revealed dynamically during the planning horizon. Each time a new customer request is received, the current plan must be re-evaluated and, in many cases, revised to accommodate the new request. The objective is to maximize the number of requests served. To address this problem, we propose a set of anticipatory scheduling policies that build on the multi-space sampling heuristic. We evaluate these policies on a new set of instances with up to 100 requests and assess their performance using a dual bound based on the perfect information relaxation. Our results show the great value of anticipation in both offline and real-time routing decisions. We also provide a comprehensive analysis of the problem parameters leading to valuable insights.

4.1 Introduction

In many countries, public utilities are responsible for providing essential services such as water, gas, electricity, and internet access. Ideally, these services should be provided without any interruption to millions of people. To ensure this, utilities perform on a daily basis a wide range of tasks at customer locations such as preventive maintenance, repairs, and meter readings. Some of these tasks are related to unforeseen problems (e.g., the wi-fi stopped working, the electricity is down, the water is not running). As a result, even if some requests are scheduled beforehand, many requests are received dynamically throughout the day (i.e., on demand). Due to the high number of customers simultaneously requesting service, utilities often lack the resources to serve all scheduled and on-demand requests on the same day. Thus, they sometimes outsource some of these requests to third-party contractors. Another difficulty faced by utilities is the inconvenient location of many customers. Indeed, in most cases, customers are located in areas with restricted access to parking. Vehicles transiting in these areas are subject to traffic delays, mixed traffic flow, or collisions. In contrast, pedestrians transit mostly without unexpected interruptions. Therefore, utilities often design vehicle routes following a park-and-loop structure (Cabrera et al. 2022; Coindreau et al. 2019; Parragh and Cordeau 2017). These routes involve a main tour that is completed using a vehicle and subtours that are carried out on foot after parking the vehicle. This leads to the dynamic park-and-loop routing problem (DPLRP).

The DPLRP is a sequential decision problem where acceptance and routing decisions are made continuously in response to, and in anticipation of, new customer requests. Solving this problem poses two big challenges. The first involves constructing an initial routing plan that accommodates all scheduled requests while maintaining sufficient flexibility to incorporate most of the on-demand requests. This initial plan must strike a delicate balance between efficiency and adaptability, ensuring that it can handle on-demand requests without substantial disruptions.

Initial routing plans are typically designed using heuristics or exact methods that minimize travel distance or time and are built offline. Flexibility can be built into the initial plan in different ways. For instance, one can design route plans that are based on both the scheduled requests and a set of sampled dynamic requests (Bent and Van Hentenryck 2004; Zhang et al. 2023). Another stream of research has focused on the design of waiting strategies (Bent and Van Hentenryck 2007) according to which workers may wait at their current location after serving a request.

The second challenge lies in the online adjustment of the initial routing plan whenever a new customer request is accepted. Updating the routing plan dynamically requires an algorithm that not only re-evaluates the current routes but also integrates new requests seamlessly. In the literature, routing plans are typically updated by inserting the new request into one of the routes, without rearranging the accepted requests or reassigning the requests among workers (Ulmer et al. 2019, 2018; Zhang and Van Woensel 2023). As a result, routing plans can be updated very fast. However, opportunities to anticipate and make room for incoming requests may be lost. To take advantage of these opportunities, researchers have also explored using more complex methods based on metaheuristics (Chen and Xu 2006; Gendreau et al. 1999). These algorithms are usually tailored to the application at hand. In the DPLRP, updating the routing plan may involve a wide range of actions, such as extending or shortening an existing walking subtour, inserting or removing a walking subtour, reassigning requests between workers, and reordering the requests in a route, among others.

To tackle these challenges, we propose a set of myopic and anticipatory scheduling policies based on a state-of-the-art metaheuristic known as the multi-space sampling heuristic (MSH), a two-phase approach that follows the route-first, assemble-second principle. Each time a new customer request is revealed, this method generates a set of high-quality routes that consider the current locations of all workers. A new solution is then formed by selecting from these routes using a set partitioning model. Although the MSH was originally a tailored method to solve the vehicle

routing problem with stochastic demands (Mendoza and Villegas 2013), thanks to its flexibility and efficiency it has been extended to other challenging vehicle routing problems. For instance, the MSH has been successfully adapted to the vehicle routing problem with stochastic travel and service times (Gómez et al. 2015), the combined maintenance and routing problem (Fontecha et al. 2016), the green vehicle routing problem (Montoya et al. 2016), the multi-period electric vehicle routing problem (Echeverri et al. 2019), and the doubly open park-and-loop routing problem (Cabrera et al. 2022).

The contributions of this paper are the following. First, we introduce and solve the dynamic park-and-loop routing problem. This problem extends the park-and-loop routing problem by considering both scheduled and on-demand requests. Second, we develop a set of scheduling policies that leverage the multi-space sampling heuristic. In contrast to other scheduling policies in the literature, our best policy allows for inter-route operations. Third, we describe a method to compute a bound on the number of requests that can be served under the assumption of perfect information. Fourth, we propose a new set of instances using the street network of Vienna, Austria. These instances are derived from the testbed introduced by Zhang et al. (2023). We make the instances, our results, and a solution checker publicly available to foster future research on this topic.

This document is organized as follows. Section 4.2 reviews the related literature. Section 4.3 formally defines the DPLRP and introduces a route-based sequential decision process. Section 4.4 describes in detail our set of scheduling policies. Section 4.5 defines the perfect information bound problem and discusses a method to solve it. Section 4.6 presents the computational experiments. Finally, Section 4.7 presents the conclusions and outlines potential paths for future research.

4.2 Literature review

In this section, we first review the literature on problems related to the DPLRP. This is followed by an overview of solution methods in Section 4.2.2.

4.2.1 Related problems

The DPLRP is closely related to the truck-and-trailer routing problem (TTRP) originally introduced by Semet (1995). In the TTRP, a fleet of trucks pulling trailers is used to serve a pre-determined set of customers, some of which can only be reached by truck without the trailer. For this reason, a subset of customers are set as decoupling points (i.e., parking places) where the trailer can be detached from the truck. The truck can then visit customers with accessibility constraints following a park-and-loop structure. Because all customers are assumed to be known beforehand, the TTRP is typically solved using static algorithms, which may be heuristic (Derigs et al. 2013; Villegas et al. 2013) or exact (Belenguer et al. 2016; Parragh and Cordeau 2017; Rothenbächer et al. 2018). These algorithms provide a routing plan that remains unchanged during the planning horizon. In contrast to the TTRP, in the DPLRP the routing plan can be updated every time a new request is revealed.

Another related problem is the truck and trailer routing problem with stochastic demand (DTTRP) introduced by Maghfiroh and Hanaoka (2018). This problem extends the TTRP by considering the random arrival of on-demand requests. To solve this problem, the authors proposed a modified simulated annealing algorithm with variable neighborhood search. This method builds an initial solution based on static information which is later modified by randomly choosing a local search operator (e.g., swap, insertion, 2-opt). To account for stochastic information, the method samples scenarios of yet-to-serve customer demand to estimate the impact of each move on the objective function. The authors test their algorithm on a set of instances with up to 70 customers. Unlike in the DTTRP, in our problem the

vehicle can be parked at any customer. As a result, the decision of where and when to park the vehicle is more complex.

The DPLRP is obviously related to the (static) park-and-loop routing problem (PLRP). This optimization problem consists of designing routes to serve a set of customers considering that workers can either walk or drive to their next location. This problem has recently received substantial attention from the research community. Coindreau et al. (2019) proposed a variable neighborhood search algorithm and a fast insertion mechanism. Cabrera et al. (2022) presented a metaheuristic based on the multi-space sampling heuristic. Le Colleter et al. (2023) introduced a small and large neighborhood search heuristic. More recently, Cabrera et al. (2023) introduced the first exact method based on the branch-price-and-cut framework capable of solving most of the existing instances with 50 customers. In the PLRP all requests are assumed to be known before the start of the planning horizon.

Finally, another related problem is the dynamic vehicle routing problem (DVRP). The DVRP consists of finding optimal vehicle routes to complete geographically dispersed deterministic and stochastic requests. There is a significant body of literature addressing the DVRP (Bent and Van Hentenryck 2007; Bent and Van Hentenryck 2004; Ferrucci et al. 2013; Sarasola et al. 2016; Ulmer et al. 2018). Bent and Van Hentenryck (2004) proposed a multiple scenario approach that consists of storing a pool of routes that resemble the current routing plan. This pool is continuously updated by performing local search on the routes that belong to the routing plan and by considering possible future requests. Bent and Van Hentenryck (2007) introduced a set of waiting and relocation strategies that allow vehicles to either wait at their current location or relocate to arbitrary sites. Ferrucci et al. (2013) introduced a real-time control approach that leverages a tabu search algorithm with the goal of guiding vehicles to request-likely areas. Sarasola et al. (2016) proposed a variable neighborhood search algorithm that adjusts the routing plan using current and sampled requests. More recently, Ulmer et al. (2018) introduced the anticipatory time budgeting heuristic, a method that relies on performing multiple simulations to

approximate an optimal decision policy, subject to the available time. This method outperformed the state-of-the-art heuristics. As opposed to the DPLRP, the DVRP only considers one type of transportation mode (i.e., driving).

4.2.2 Solution methods

To solve dynamic vehicle routing problems, researchers have explored many alternatives. Arguably the most common method is known as *reoptimization*. Reoptimization updates the current routing plan by solving a *static* version of the problem with the most recent available information. Typically, this is done by using metaheuristics, as solutions are to be obtained in relatively short computational times. Depending on how often the static problem is solved, reoptimization methods can be classified into periodic (PR) and continuous (CR). As the name suggests, PR methods solve the static problem at fixed intervals of time (e.g., every 5 minutes) or after observing a pre-defined number of events (e.g., after five customer requests are received). For example, Pillac et al. (2018) developed a fast metaheuristic that is applied every time a new customer request is revealed. Alternatively, CR methods run continuously during the planning horizon. To do so, these methods often rely on maintaining a pool of solutions that can be adapted to the newly disclosed information without significant effort. For instance, Gendreau et al. (1999), presented a parallel tabu search to continuously improve the set of feasible solutions. This method is executed in multiple threads that are only interrupted when new information is available.

Another line of research models dynamic vehicle routing problems as Markov decision processes (MDPs). An MDP is well-suited for modeling as it explicitly incorporates uncertainty and stepwise planning (Ulmer 2017). However, these models face the well-known *curse of dimensionality* (Powell 2007), as real-world problems involve vast state, action, and outcome spaces, rendering exact solution methods computationally infeasible. As a result, researchers have focused on the design of

approximate dynamic programming (ADP) methods. The simplest ADPs are myopic policies, which, as the name suggests, ignore future customer requests. Instead, they focus solely on maximizing immediate rewards, often resulting in poor performance. To overcome this, more complex methods that explicitly sample and simulate dynamic requests have been developed. These methods can be classified into three categories: rollout algorithms, policy function approximation (PFA), and value function approximation (VFA).

Rollout algorithms, introduced by Bertsekas et al. (1997), leverage an existing policy, known as the *base policy*, to improve decision-making in dynamic programming problems. The intuition behind these methods is to *roll out* the base policy by simulating multiple future steps starting from the current state. These simulations generate potential outcomes for different actions, enabling an evaluation of their quality based on the expected cumulative reward over the simulated horizon. For instance, Secomandi (2001) developed a rollout algorithm for the vehicle routing problem with stochastic demands, using the cyclic heuristic as the base policy. Their results demonstrated performance improvements of up to 4% on instances with up to 40 customers. However, a key limitation of rollout algorithms is that all simulations are performed online, which can result in significant delays in real-time decision-making. In this paper, we use a rollout algorithm as one of our benchmark algorithms.

In a search of efficiency, researchers have developed policy function approximation methods (Ulmer and Streng 2019; Ulmer and Thomas 2018). These methods use decision rules designed to emulate effective decision-making, typically relying on thresholds determined through offline simulations. For instance, Ulmer and Thomas (2018) proposed a PFA for solving the dynamic vehicle routing problem with heterogeneous fleets of drones and vehicles. Their approach divides the service area into two zones based on vehicle travel time from the depot. Customers within a specified threshold distance are served by vehicles, while those beyond it are served by drones. Similarly, Ulmer and Streng (2019) designed a PFA for same-day delivery systems

involving pickup stations and autonomous vehicles. This method determines when and where to dispatch vehicles to pickup stations and which goods to load, using a threshold to balance fast delivery with consolidation efficiency. Since the thresholds are pre-defined, PFA methods are highly efficient. However, they are only suitable for problems in which a good policy can be readily observed (Powell 2007).

Value function approximation is another common approach (Soeffker et al. 2024; Ulmer et al. 2019, 2020; Ulmer and Thomas 2020). This method estimates the value of being in a particular state or the expected value of taking a specific action. Due to the typically large number of states, they are often aggregated into a set of features. Ulmer et al. (2018) proposed a non-parametric value function based on the available time to approximate the cost of routing decisions. Their solution incorporates a dynamic lookup table that adaptively partitions the state space during the learning process. Soeffker et al. (2024) introduced parametric VFAs that model vehicle workloads at varying levels of detail, enabling effective customer request assignments. Similarly, Ulmer and Thomas (2020) developed a meso-parametric VFA (M-VFA) that integrates non-parametric and parametric VFAs, leveraging their strengths while mitigating their limitations. In a related effort, Ulmer et al. (2019) combined a VFA with a rollout algorithm to address the dynamic vehicle routing problem with stochastic requests. This hybrid approach demonstrated the benefits of blending offline and online methods to generate higher-quality policies.

The primary drawback of VFAs lies in their high computational complexity, as these methods often rely on lookup tables that are susceptible to the curse of dimensionality. To address this issue, Zhang et al. (2023) proposed a set of knapsack-based linear models that compute the expected reward of a decision without requiring offline simulations. In these models, each vehicle is represented as a knapsack, where the capacity corresponds to the remaining service time available in the vehicle's shift. By combining these models with an intra-route routing policy, they developed efficient online scheduling policies. Under these policies, once a request is assigned to a vehicle, it cannot be reassigned. Their experiments on large-scale,

real-time instances demonstrated that these policies outperformed traditional ADP approaches in both solution quality and computational efficiency. Building on this concept, we adapt the knapsack-based linear models and propose scheduling policies that leverage the multi-space sampling heuristic to enable efficient inter-route operations. Our experiments indicate that our scheduling policies enhance solution quality with minimal impact on computational efficiency.

Table 4.1 presents a classification of the literature related to this paper. Column 1 provides a reference for each article. Columns 2 and 3 indicate whether or not the article considers routing and parking decisions, respectively. Column 4 shows if the article considers uncertainty in the problem parameters, such as the travel times (T) or the demand (D). Column 5 shows if the method proposed in the article anticipates the impact of a decision in the future costs or rewards. Column 6 reports the method used to build the initial solution. Column 7 shows the method that updates the initial solution.

4.3 Problem definition

In this section, we formally define the DPLRP. We first present the problem description and we provide an illustrative example. We then present a route-based sequential decision process inspired by the route-based Markov decision processes introduced by Ulmer et al. (2020).

4.3.1 Problem description

The DPLRP can be formally defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the node set and $\mathcal{A}$ is the arc set. The set $\mathcal{N}$ comprises the depot represented by node $\overline{0}$ and the customer request set $\mathcal{C}$. Each customer request $i \in \mathcal{C}$ has a fixed duration ν_i. To serve customer requests, a set $\mathcal{W}$ of homogeneous workers is available. Arcs in $\mathcal{A}$ represent the connections between two locations. Each arc (i, j) has four main

Table 4.1: Literature classification.

Author	Decisions		Uncertainty	Anticipation	Offline policy	Online policy
	Routing	Parking				
Chao (2002)	✓	✓			TS	n/a
Derigs et al. (2013)	✓	✓			LNS	n/a
Villegas et al. (2013)	✓	✓			GRASP	n/a
Belenguer et al. (2016)	✓	✓			B&C	n/a
Parragh and Cordeau (2017)	✓	✓			BAP	n/a
Rothenbächer et al. (2018)	✓	✓			BPC	n/a
Coindreau et al. (2019)	✓	✓			VNS	n/a
Martinez-Sykora et al. (2020)	✓	✓			B&C	n/a
Cabrera et al. (2022)	✓	✓			MSH	n/a
Cabrera et al. (2023)	✓	✓			BPC	n/a
Le Colleter et al. (2023)	✓	✓			SLNS	n/a
Zhou et al. (2023)	✓	✓			BAP	n/a
Cabrera et al. (2025)	✓	✓			BPC	n/a
Gendreau et al. (1999)	✓		D		TS	TS
Bent and Van Hentenryck (2004)	✓		D	✓	MSA	MSA
Mitrović-Minić and Laporte (2004)	✓		D		n/a	TS
Bent and Van Hentenryck (2007)	✓		D	✓	LNS	LNS
Ferrucci et al. (2013)	✓		D		TS	TS
Ulmer et al. (2018)	✓		D	✓	CI	PFA+rollout
Maghfiroh and Hanaoka (2018)	✓	✓	D		CI	SA
Pillac et al. (2018)	✓		D		ALNS	ALNS
Ulmer et al. (2019)	✓		D	✓	CI	VFA+rollout
Voccia et al. (2019)	✓		D	✓	n/a	MSA
Ulmer and Thomas (2020)	✓		D	✓	n/a	VFA
Ulmer et al. (2020)	✓		D	✓	n/a	VFA
Zhang and Van Woensel (2023)	✓		D	✓	PB	PB
This work	✓	✓	D	✓	PB-MSH	PB-MSH

attributes: the walking distance λ_{ij}, the driving distance μ_{ij}, the walking time γ_{ij}, and the driving time δ_{ij}. Every time a worker arrives at a location using a vehicle, there is an associated parking time τ. The maximum walking distance between two points is Υ and the maximum duration of a walking subtour is Γ.

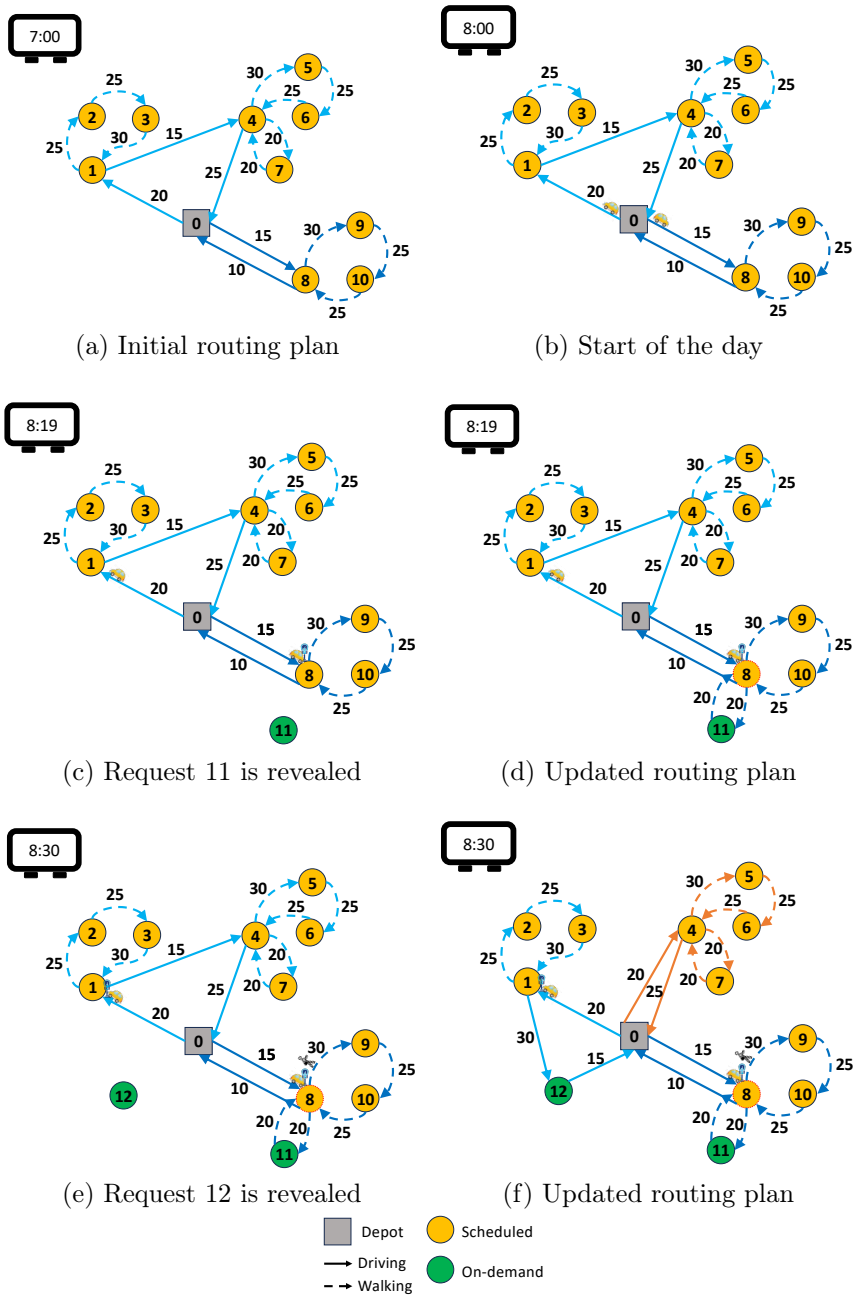
Set $\mathcal{C}$ can be divided into two disjoint subsets. The first subset contains the scheduled requests $\mathcal{C}^s$, which are known at the beginning of the planning horizon. These requests must be served before the end of the planning horizon. The second set contains the on-demand requests $\mathcal{C}^d$, which are not known in advance. We assume these requests appear randomly during the working hours $[e, f]$. For each new customer request, a decision is made to either accept or reject it. If the request is accepted, it must be served within the planning horizon. On the contrary, if rejected, it is assumed to be outsourced or postponed to a later date.

We also define a route plan θ as a set of park-and-loop routes $\{r_1, \ldots, r_{\mathcal{W}}\}$ for

each worker. A park-and-loop route $r = \{\overline{0}, i_1, i_2, \ldots, i_{|r|-2}, \overline{0}\}$ is a sequence of nodes that starts and ends at the depot. The objective of the DPLRP is to design a route plan that maximizes the number of on-demand requests served such that: the total duration of all routes performed by any worker does not exceed the workday duration; the walking distance for each worker does not exceed the walking distance limit; and the total duration of a waking subtour does not exceed the time limit.

Figure 4.1 shows an example of an instance of the DPLRP with three workers. In this instance, the service time of every customer request is ten minutes and the time limit for a route is seven hours. Also, the parking time is set to two minutes. The driving or walking times between each location are indicated next to each arc. Figure 4.1a shows the initial routing plan composed of two park-and-loop routes. This routing plan is built before the start of the workday. According to the initial routing plan, Worker 1 (light blue) departs at 8:00 am from the depot towards Request 1. There, the worker parks the vehicle before serving the request. Later, the worker walks to serve requests 2 and 3, before going back to retrieve the vehicle. Afterward, the worker drives to Request 4. After parking the car and serving the request, the worker walks to serve requests 5 and 6. Then, after going back to the parking spot, the worker walks to Request 7. Finally, after serving the request and retrieving the vehicle, the worker returns to the depot. Similarly, Worker 2 (dark blue) is scheduled to drive towards Request 8. Then, the worker walks to serve requests 9 and 10. Finally, after retrieving the vehicle, the worker drives to the depot.

Figure 4.1b shows the park-and-loop routes being executed at the start of the planning horizon. Figure 4.1c shows the park-and-loop routes being executed at 8:19 am, right after Request 11 appears. At 8:19 am, Worker 1 is about to reach the location of Request 1, while Worker 2 is already servicing Request 8. Due to the arrival of a new request and the proximity of Worker 2 to Request 11, the initial routing plan is modified as shown in Figure 4.1d. In particular, a new subtour is added to the route of Worker 2 to ensure that Request 11 will be served. The route



(a) Initial routing plan (b) Start of the day

(c) Request 11 is revealed (d) Updated routing plan

(e) Request 12 is revealed (f) Updated routing plan

Figure 4.1: Illustrative example of an instance of the DPLRP.

of Worker 1 remains unchanged. At 8:30 am, a new on-demand request is unveiled, namely, Request 12 as shown in Figure 4.1e. To add this request to the routing plan, the route of Worker 1 is slightly changed and a new route is created (dark orange) as shown in Figure 4.1f. In addition, Worker 1 is now scheduled to serve Request

12.

This example illustrates the challenges and complexity of the DPLRP. When a new customer request appears, it is crucial to decide whether to accept or reject it. While accepting a new customer request may seem advantageous at the time, it could prevent the acceptance of two or more subsequent requests. Additionally, if the customer request is accepted, the routing plan must be adjusted to accommodate it. For instance, a subtour could be created within an existing route, or an existing subtour could be shortened or extended. Another option is to reassign requests among workers. This option could drastically change each route in the routing plan and could also involve creating new routes for idle workers at the depot. A solution method for the DPLRP must consider all these aspects and be able to quickly provide a solution.

4.3.2 Route-based sequential decision process

In this section, we present a route-based sequential decision process that captures the different forms of uncertainty in the DPLRP.

Decision epochs

Decision epochs $k \in \{0, \ldots, \mathcal{K}\}$ refer to specific points in time when decisions are made. In the DPLRP, a decision is made every time that an on-demand request i_k appears. In addition, a decision is made at the start of the planning horizon to design the initial routing plan. The number of decision points per day $\mathcal{K}$ is a random variable.

States

A state represents the current status of the system at a specific decision epoch k. We define the state of the system at decision epoch k as $\mathcal{S}_k$. In our problem, the state $\mathcal{S}_k$ consists of the following eight components:

- the current time denoted by $t_k \in \mathbb{R}_{\geq 0}$,
- the current routing plan denoted by θ_k,
- the current destination of each worker w denoted by $\hat{\imath}_{wk} \in \mathcal{N}$,
- the arrival time of each worker w at their current destination denoted by $\hat{\jmath}_{wk} \in [e, f]$,
- the available time that each worker w has in their current walking subtour denoted by $\hat{u}_{wk} \in [0, \Gamma]$,
- the distance that can still be walked by each worker w denoted by $l_{wk} \in [0, \Upsilon]$,
- the current parking spot of the vehicle of each worker w denoted by $p_{wk} \in \mathcal{N}$,
- and the set of accepted but not yet served requests denoted by $\mathcal{C}_k$.

State $\mathcal{S}_k$ can be summarized as $\mathcal{S}_k = (t_k, \theta_k, \hat{\imath}_k, \hat{\jmath}_k, \hat{u}_k, l_k, p_k, \mathcal{C}_k)$ with $\hat{\imath}_k$, $\hat{\jmath}_k$, $\hat{u}_k$, l_k, and p_k being $|\mathcal{W}|$-dimensional vectors. If a worker is serving a request, the value of $\hat{\imath}_k$ is set to the worker's current location. In addition, if a worker is not performing a subtour at time t_k, the value of $\hat{u}_{wk}$ is set to 0. Also, if the worker is driving between two locations, p_{wk} is set to the current destination. In the initial state $\mathcal{S}_0 = (0, \theta_0, \hat{\imath}_0, \hat{\jmath}_0, \hat{u}_0, l_0, p_0, \mathcal{C}_0)$, all workers are idle at the depot and $\mathcal{C}_0 = \mathcal{C}^s$. To guarantee feasibility, we assume that there exists a routing plan that can serve all the requests in $\mathcal{C}^s$. At the final decision epoch $\mathcal{K}$, the process is at a terminal state $\mathcal{S}_\mathcal{K} = (t_\mathcal{K}, \theta_\mathcal{K}, \hat{\imath}_\mathcal{K}, \hat{\jmath}_\mathcal{K}, \hat{u}_\mathcal{K}, l_\mathcal{K}, p_\mathcal{K}, \varnothing)$, where all the workers are back at the depot and all accepted requests are served.

Actions

Actions are the decisions made at each decision epoch k that influence the state of the system. At state $\mathcal{S}_0$ the set of feasible decisions $\mathcal{F}$ is composed of all the feasible routing plans θ that include the set of scheduled requests $\mathcal{C}^s$. We represent

the initial decision by $\theta_0 \in \mathcal{F}$. This decision is made offline prior to the start of the planning horizon. For the remaining decision epochs $k \neq 0$, decisions are made online. Given $\mathcal{S}_k$, an acceptance decision is made about whether to accept or reject the new on-demand request i_k. In case the new on-demand request is accepted, it is also necessary to make a routing decision according to which the routing plan θ_k is updated. There are only two possible acceptance decisions, but there are many possible routing decisions.

We denote by $\mathcal{X}_k$ the set of feasible decisions at state $\mathcal{S}_k$. Each decision $x_k \in \mathcal{X}_k$ specifies both the acceptance and the routing decisions. If the new on-demand request is rejected, the routing decision defaults to preserving the current routing plan. Formally, a decision is a tuple $x_k = (\mathbb{I}_k^x, \theta_k^x)$, where $\mathbb{I}_k^x$ is an indicator variable that takes the value of 1 if the on-demand request i_k is accepted and θ_k^x is the updated routing plan that must include all the request that are yet to be served: $C_k = C_k \cup \{i_k\}$. A decision is feasible when the following conditions hold:

- the park-and-loop routes $r \in \theta_k^x$ are feasible,
- the workers driving or walking between two locations pursue their current destinations,
- and the accepted request is assigned to exactly one worker.

Transitions

When a decision x_k is made at decision epoch k, it leads to the state transition from the pre-decision state $\mathcal{S}_k$ to post-decision state $\mathcal{S}_k^x = (t_k^x, \theta_k^x, \hat{\imath}_k^x, \hat{\jmath}_k^x, \hat{u}_k^x, l_k^x, p_k^x, C_k^x)$. In $\mathcal{S}_k^x$ we update the workers' destinations and arrival times of the previously idle workers. These values are inherited from the updated routing plan θ_k^x. Similarly, we update the available time for performing a walking subtour and the remaining distance to be covered on foot. We also update the current parking spots for each worker. Finally, the set of accepted but not yet served requests is updated by remov-

ing the requests served between decision epochs $k-1$ and k, and by adding request i_k in case an accept decision was made. The transition from the post-decision state $\mathcal{S}_k^x$ to the next pre-decision state $\mathcal{S}_{k+1} = (t_{k+1}, \theta_{k+1}, \hat{\imath}_{k+1}, \hat{\jmath}_{k+1}, \hat{u}_{k+1}, l_{k+1}, p_{k+1}, C_{k+1})$ occurs at the time of the $(k+1)$-th decision epoch t_{k+1}. The transition updates the state as follows:

- the routing plan θ_{k+1} is obtained by removing the requests served before t_{k+1} from θ_k^x,
- the set of accepted but not yet served requests is modified as $C_{k+1} = C_k^x \setminus \tilde{C}$, where $\tilde{C}$ corresponds to the request served before time t_k,
- and a new on-demand request i_{k+1} appears.

Rewards

The reward quantifies the immediate benefit (or cost) of an action. We describe the reward of a decision x_k as a marginal reward that accounts for the difference between the value of a previous route plan and the new one. Let $CR(\theta)$ be the reward obtained by route plan θ. Also, let $\tilde{\mathbb{I}}_k$ be an indicator variable that takes the value of 1 if the on-demand request i_k was accepted. Then, the reward reflecting the accepted requests at any decision epoch k is defined as:

$$CR(\theta_k) = \sum_{i=0}^{k-1} \mathbb{I}_i + |C^s|. \tag{4.1}$$

This equation considers both the scheduled requests and the previously accepted on-demand requests. After decision x_k is taken at decision epoch k, the reward of the updated routing plan θ_k^x can be written as:

$$CR(\theta_k^x) = CR(\theta_k) + \mathbb{I}_k^x. \tag{4.2}$$

Then, the reward $R(\cdot)$ of a decision x_k is defined as:

$$R(\mathcal{S}_k, x_k) = CR(\theta_k^x) - CR(\theta_k). \tag{4.3}$$

Objective

Let Π denote the set of Markovian deterministic scheduling policies, where a policy $\pi \in \Pi$ is a sequence of decision rules $(X_0^\pi, X_1^\pi, \ldots, X_{\mathcal{K}}^\pi)$ to map a state $\mathcal{S}_k$ to decision $x_k = X_k^\pi(\mathcal{S}_k)$. These decision rules may be identical for every epoch $k \in \{1, \ldots, \mathcal{K}\}$. We seek an optimal scheduling policy $\pi^* \in \Pi$ that maximizes the expected total reward beginning from the initial state $\mathcal{S}_0$:

$$V(\mathcal{S}_0) = \max_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{\mathcal{K}} R(\mathcal{S}_k, X_k^\pi(\mathcal{S}_k)) | \mathcal{S}_0\right]. \tag{4.4}$$

In the context of the DPLRP, a scheduling policy comprises an offline policy that defines the initial routing plan θ_0, an acceptance policy that decides whether to accept or reject a request, and a routing policy that updates the routing plan after the acceptance decisions. In the following section, we present different alternatives for each of these policies.

4.4 Scheduling policies

In the following, we present an overview of the multi-space sampling heuristic. Then, we define the routing, acceptance, and offline policies.

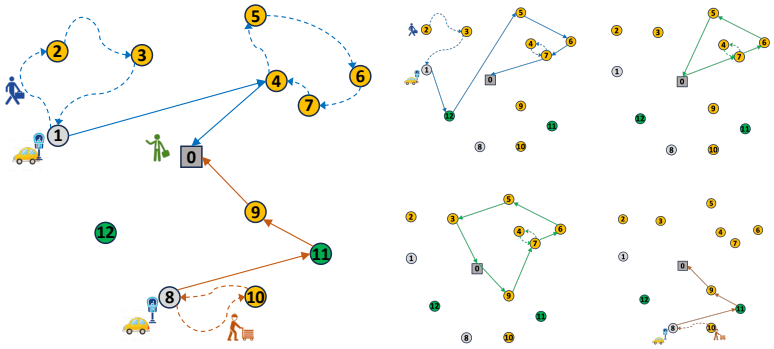
4.4.1 Multi-space sampling heuristic

The MSH comprises two phases, namely, a route generation and an assembly phase. In the route generation phase, the algorithm iteratively builds a pool (i.e., a set) of

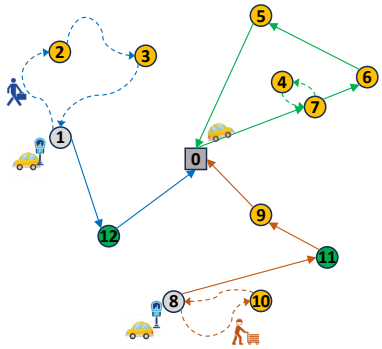
high-quality routes by using a set of route generation functions. A route generation function can for example take the form of a local search procedure that improves an existing route. In the assembly phase, the algorithm solves a set partitioning model that builds a final solution. This step is carried out by an assembler. Figure 4.2 illustrates the two phases of MSH applied to an instance of the DPLRP with ten scheduled requests and two on-demand requests. Figure 4.2a depicts the current state of the system: one worker, colored in green, is idle at the depot, while the other two workers, colored in blue and orange, are walking toward Requests 2 and 10, respectively. Figure 4.2b shows a selection of routes generated during the first phase of the MSH, with each route beginning from the workers' current destinations or, in the case of the idle worker, their current location. Finally, Figure 4.2c displays the updated routing plan. In this new routing plan, the worker in green is assigned to depart immediately toward Request 7.

Algorithm 11 presents the main logic of the MSH. Line 1 initializes the pool of park-and-loop routes $\overline{\Omega}_w$ for each worker $w \in \mathcal{W}$. Then, we enter the outer loop (lines 2-9) where the algorithm populates each pool of routes by iterating over a set of route generation functions Ξ. This outer loop is executed in parallel by ζ independent threads. Line 3 initializes the iteration number. Afterward, the algorithm uses the inner loop (lines 5-7) for a given number of iterations (Q) or a maximum execution time (Ψ). Line 5 computes a subset of routes r^q using the selected route generation function. Line 6 adds the routes found to the corresponding pool of routes. Line 10 assembles a solution composed of one or multiple routes by calling the assembler. This solution is stored in $\tilde{\Omega}_w$. Finally, Line 11 returns the final solution $\tilde{\Omega}_w$.

Depending on the definition of the set of route generation functions Ξ and the assembler, different scheduling policies emerge. In what follows, we provide a general definition of the assembler. A more detailed discussion and definition of the route generation functions is deferred to Sections 4.4.3 and 4.4.3. The assembler can be generally defined as follows. Let Ω_w^k define a subset of feasible park-and-loop routes



(a) Current routing plan (b) Subset of feasible routes



(c) Updated routing plan

Figure 4.2: Multi-space sampling heuristic example.

Algorithm 11 `MSH` function

Require: $\mathcal{G}$, graph; Ξ, route generation functions; Q, iteration limit; Ψ, time limit route generation phase; Φ, time limit assembly phase.

Ensure: final solution $\tilde{\Omega}_w$

$\overline{\Omega}_w \leftarrow \emptyset$
for $\xi \in \Xi$ **do**
  $q \leftarrow 1$
  while $q < \frac{Q}{|\Xi|} \wedge route\_generation\_time < \Psi$ **do**
    $r^q \leftarrow \xi(\mathcal{G})$
    $\overline{\Omega}_w \leftarrow \overline{\Omega}_w \cup r^q$
    $q \leftarrow q + 1$
  end while
end for
$\tilde{\Omega}_w \leftarrow \texttt{Assembly}(\mathcal{G}, \overline{\Omega}_w, \Phi)$
return $\tilde{\Omega}_w$

for each worker given their status at decision epoch k. Let h_r be the cost of route r. Also, let a_{ir} be a binary parameter that takes the value 1 if route r serves request i, and 0 otherwise. Let y_r be a binary variable that takes the value 1 if route r is selected, and 0 otherwise. Then, the assembler at any decision epoch $k > 0$ is defined as:

$$\min \sum_{w \in \mathcal{W}} h_r y_r \tag{4.5}$$

subject to

$$\sum_{w \in \mathcal{W}} \sum_{r \in \Omega_w^k} a_{ir} y_r = 1 \qquad \forall i \in C_k \tag{4.6}$$

$$\sum_{w \in \mathcal{W}} \sum_{r \in \Omega_w^k} a_{ir} y_r \leq 1 \qquad \forall i = i_k \tag{4.7}$$

$$\sum_{r \in \Omega_w^k} y_r = 1 \qquad \forall w \in \mathcal{W} \tag{4.8}$$

$$y_r \in \{0, 1\} \qquad \forall w \in \mathcal{W}, r \in \Omega_w^k. \tag{4.9}$$

The objective function (4.5) minimizes the total cost of the routing plan. Constraints (4.6) ensure that all the scheduled and previously accepted requests are included in the routing plan. Constraints (4.7) state that if accepted, the new on-demand request can only be assigned to one route. Constraints (4.8) guarantee that one route is assigned to each worker. Formulation (4.5)-(4.9) can be easily adapted to handle additional constraints. Moreover, the definition of the cost h_r for each route can be redefined to accomplish different goals. For example, one could consider the total travel time of a route as its cost, which would likely result in selecting shorter routes.

4.4.2 Acceptance policies

An acceptance policy determines whether or not to accept a new customer request. A straightforward approach to maximize the number of served requests is to employ

a greedy acceptance policy ϱ_G, which accepts customer requests whenever feasible. Formally, the greedy acceptance policy accepts every customer request if it can be integrated into the current routing plan using one of the routing policies described in Section 4.4.3. However, the greedy policy does not account for the potential arrival of future on-demand customer requests, leading to suboptimal performance in many scenarios. In the following, we define an acceptance rule and we introduce two acceptance policies.

Acceptance rule

We define the expected reward-to-go at state $\mathcal{S}_k$ when using scheduling policy π as:

$$\Phi_\pi(\mathcal{S}_k) = \mathbb{E}\left[\sum_{i=k+1}^{\mathcal{K}} R(\mathcal{S}_i, X_i^\pi(\mathcal{S}_i))|\mathcal{S}_k\right]. \tag{4.10}$$

Then, the acceptance rule at any decision epoch k is defined as:

$$\mathbb{I}_k^x = \mathbb{I}(\Phi_\pi(\mathcal{S}_k^+) \geq \Phi_\pi(\mathcal{S}_k^-)), \tag{4.11}$$

where $\mathcal{S}_k^+$ and $\mathcal{S}_k^-$ are the post-decision states corresponding to accept and reject decisions, respectively. Equation (4.11) states that a customer request is accepted if and only if the expected reward obtained after accepting the request is higher than the expected reward obtained if the request is rejected. Computing the expected reward of a decision is not easy. For this reason, most researchers have turned to approximations based on simulation (Ulmer et al. 2019, 2018; Zhang et al. 2023). For the sake of conciseness, we define a set of scenarios Σ in which each scenario $\sigma \in \Sigma$ represents a sample path of the customer request arrival process from a given epoch $k \in \mathcal{K}$ to the end of the planning horizon. In what follows we describe our adaptation of the multi-knapsack approximation introduced by Zhang et al. (2023) that is used to build a potential-based acceptance policy. We also present a rollout acceptance policy.

Multi-knapsack approximation

The multi-knapsack approximation of the expected reward was introduced by Zhang et al. (2023) in the context of the DVRP. The intuition behind this approximation is to quickly assess how many future customer requests can be potentially accepted considering the current routing plan θ_k. To do so, it uses the set of scenarios Σ. Let C_σ represent the on-demand requests that arrive in scenario σ and let η_{ir} be the increase in the duration of a route if request $i \in C_\sigma$ is inserted into route $r \in \theta_k$. Also, let z_{ir} be a binary variable that takes the value 1 if the request $i \in C_\sigma$ is inserted into route $r \in \theta_k$, and 0 otherwise. Then, the multi-knapsack approximation model when considering scenario $\sigma \in \Sigma$ can be defined as:

$$\phi_\pi^\sigma(\mathcal{S}_{k+1}) = \max \sum_{i \in C_\sigma} \sum_{r \in \theta_k} z_{ir} \tag{4.12}$$

subject to

$$\sum_{i \in C_\sigma} \eta_{ir} z_{ir} \leq f - T_r \qquad \forall r \in \theta_k \tag{4.13}$$

$$\sum_{r \in \theta_k} z_{ir} \leq 1 \qquad \forall i \in C_\sigma \tag{4.14}$$

$$z_{ir} \in \{0, 1\} \qquad \forall i \in C_\sigma, r \in \theta_k. \tag{4.15}$$

The objective function (4.12) maximizes the number of requests inserted into the routing plan. Constraints (4.13) guarantee that the total increase in the duration of a route does not exceed the time available. Constraints (4.14) state that each on-demand request can be inserted into at most one route. Constraints (4.15) state that the variables are binary. To speed up the solution of each independent model, Zhang et al. (2023) suggest relaxing the integrality requirements on the z_{ir} variables. We follow this suggestion in our implementation. To obtain the expected reward, this model has to be solved independently for each scenario $\sigma \in \Sigma$. Then, the expected reward $\Phi_\pi(\mathcal{S}_k)$ when at state $\mathcal{S}_k$ can be approximated as:

$$\Phi_\pi(\mathcal{S}_k) \approx \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} \phi_\pi^\sigma(\mathcal{S}_{k+1}). \tag{4.16}$$

The potential-based acceptance policy $\varrho_{MK-|\Sigma|}$ consists of using the multi-knapsack approximation to evaluate the acceptance rule (4.11).

Rollout acceptance policy

As previously explained, a rollout algorithm is a method that leverages a *base policy* to improve decision-making (Bertsekas et al. 1997). To do so, it evaluates different actions by *rolling out* the base policy from the current state to the end of the planning horizon. This approach ensures that each decision is made considering its long-term impact, leading to more effective and adaptive solutions. Due to the large number of simulations required, the base policy must be highly computationally efficient. In our case, the base policy corresponds to the greedy acceptance policy ϱ_G. Formally, given a routing policy ρ and a state $\mathcal{S}_k$, the rollout policy $\varrho_{R-|\Sigma|}$ simulates the system over the set of scenarios Σ given that policy ϱ_G is in effect. That is, every time a new request appears, an acceptance decision is made using policy ϱ_G. Let $\phi^\sigma(\mathcal{S}_k^+)$ and $\phi^\sigma(\mathcal{S}_k^-)$ be the total reward obtained in scenario $\sigma \in \Sigma$ after accepting and rejecting request i_k, respectively. Then, request i_k is accepted if $\sum_{\sigma \in \Sigma} \phi^\sigma(\mathcal{S}_k^+) \geq \sum_{\sigma \in \Sigma} \phi^\sigma(\mathcal{S}_k^-)$. Otherwise, the request is rejected.

4.4.3 Routing policies

A routing policy determines how workers are mobilized to serve requests. Routing policies are usually designed to achieve specific objectives such as minimizing the total travel distance or total cost. Our first two routing policies are myopic. That is, they ignore that new customer requests may appear before the end of the planning horizon. Instead, they aim to minimize the total travel time. On the contrary, our third routing policy is anticipatory. That is, it aims to maximize the expected

number of served customer requests. In what follows, we describe these policies in detail.

Myopic best-insertion routing policy

The first routing policy relies on the best insertion route generation function $\xi^{BI}(r,i) \in \Xi$. This route generation function creates a new route r' using as input an existing route r and a request i. More precisely, route r' is computed by inserting request i in the position of the route r that minimizes the insertion cost without changing the order for the customers that are already in the route. Mathematically, this cost can be defined as follows. Let $\mathcal{A}_r^d$ be the ordered set of driving arcs used by route r. Let $\tilde{\mathcal{C}}_r$ represent the ordered set of requests in route r. Also, let T_r represent the total duration of route r. Then, the cost c_{ijl}^d of inserting request $i \in \overline{\mathcal{C}}$ between request j and request l can be computed as $c_{ijl}^d = \delta_{ji} + \delta_{il} - \delta_{jl}$, where $(j,l) \in \mathcal{A}_r^d$. If the insertion is not feasible, that is if $c_{ijl}^d + T_r + \nu_i > f$, this value is set to infinity. Note that this route generation function assumes that request i is served using the vehicle. In some cases, this may be suboptimal, as it could be beneficial to walk between requests instead.

To overcome this issue, we start by defining the cost of creating a new walking subtour departing from request $j \in \tilde{\mathcal{C}}_r$ and serving request $i \in \overline{\mathcal{C}}$ as $c_{ij}^w = \gamma_{ji} + \gamma_{ij}$. Similarly, if the insertion is not feasible, that is if $c_{ij}^w + T_r + \nu_i > f$, this value is set to infinity. Then, using this definition, we propose the route generation function $\xi^{TI}(r,i)$ that builds a route r' by creating a new walking subtour such that c_{ij}^w is minimized. In addition, we define a route generation function $\xi^{ES}(r,i) \in \Xi$ that extends an existing subtour. Let $\mathcal{U}_r$ be the set of the walking subtours in route r. A walking subtour $u \in \mathcal{U}_r$ is defined as a sequence of nodes $\{i_1, i_2, \ldots, i_1\}$ that starts and ends at the same node. The walking arcs (in order) in each subtour are defined by $\mathcal{A}_u$. Also, let $\mathcal{T}_u$ be the total duration of the walking subtour $u \in \mathcal{U}_r$. Then, we define the cost of extending a walking subtour $u \in \mathcal{U}_r$ by inserting request $i \in \overline{\mathcal{C}}$ as:

$$c^e_{iu} = \min_{(j,l)\in\mathcal{A}_u} \left\{\gamma_{ji} + \gamma_{il} - \gamma_{jl}\right\} + \nu_i. \tag{4.17}$$

If the insertion is not feasible, that is if $c^e_{iu} + \mathcal{T}_u > \Gamma$ or $c^e_{iu} + \mathcal{T}_u > f$, this value is set to infinity. Then, the route generation function $\xi^{ES}(r,i)$ builds a route r' by extending the walking subtour $u \in \mathcal{U}_r$ such that the cost of extending the walking subtour is minimized. Note that the route generation functions $\xi^{BI}(r,i)$, $\xi^{TI}(r,i)$, and $\xi^{ES}(r,i)$ do not change the order of the already accepted requests. Moreover, they do not consider accepted requests other than those already serviced by the current route.

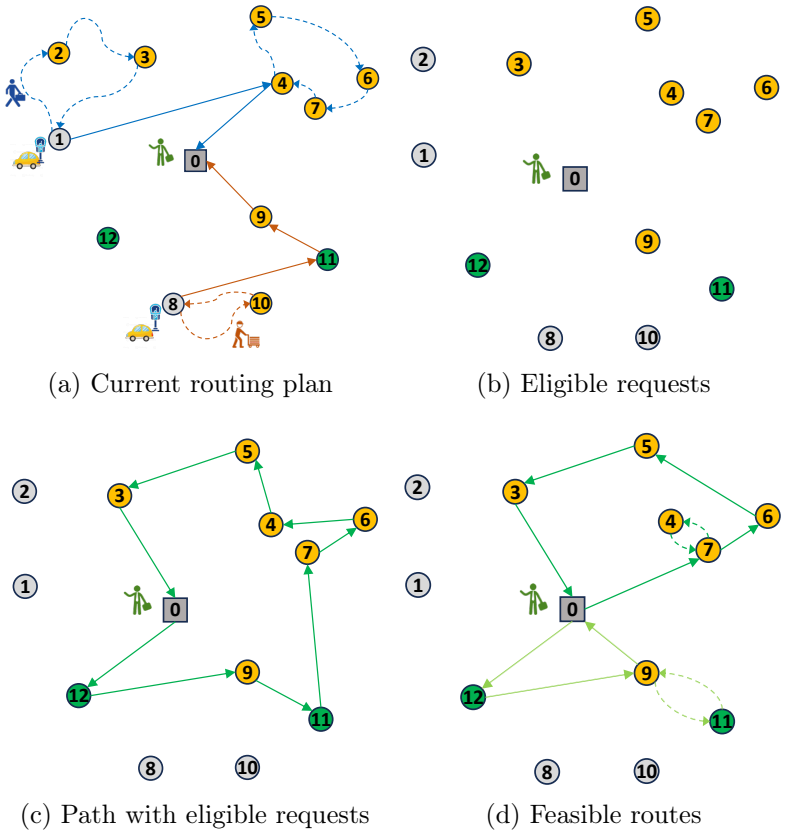
Using route generation functions $\xi^{BI}(r,i)$, $\xi^{TI}(r,i)$, and $\xi^{ES}(r,i)$ we define the myopic best-insertion routing policy ρ_{MI} in which the set of route generation functions is defined as $\Xi = \bigcup_{r\in\theta} \xi^{BI}(r,i) \cup \bigcup_{r\in\theta} \xi^{TI}(r,i) \cup \bigcup_{r\in\theta} \xi^{ES}(r,i)$. Moreover, the cost of each route in the assembler h_r is set to the total duration of the route T_r.

Myopic inter-route routing policy

Our second routing policy allows for inter-route operations. That is, it allows for the reassignment of accepted requests between workers. It also allows for more complex operations such as extending or shortening existing walking subtours, and the insertion of new ones if convenient. To do so, we define the route generation function $\xi^{\tau}(w, C_k, i_k)$. This route generation function creates a new subset of routes Ω_w for worker w using as input the set C_k of accepted but not yet served requests, and the new on-demand request i_k. The route generation function $\xi^{\tau}(w, C_k, i_k)$ relies on two components. First, we use a randomized traveling salesman problem (TSP) heuristic that creates paths from the worker's current location to the depot, including both the accepted but not yet served requests and the new on-demand request. In particular, we use four classic TSP construction heuristics, namely, randomized nearest neighbor (RNN), randomized nearest insertion (RNI), randomized farthest insertion (RFI), and randomized best insertion (RBI). Second, we use a tailored

split algorithm introduced by Cabrera et al. (2022) that partitions the path into feasible routes.

Figure 4.3 illustrates how route generation function $\xi^{\tau}(w, C_k, i_k)$ operates. Figure 4.3a presents the current state of the system, including the location of an idle worker highlighted in green. Figure 4.3b displays the eligible requests, i.e., the accepted but not served requests and the new on-demand Request 12. Figure 4.3c shows a feasible path that begins at the worker's current location (i.e., the depot) and visits all the accepted but not yet served requests, as well as the new on-demand Request 12. Finally, Figure 4.3d shows two park-and-loop routes obtained after using the split algorithm, one of which includes the new on-demand request.



(a) Current routing plan (b) Eligible requests

(c) Path with eligible requests (d) Feasible routes

Figure 4.3: Inter-route route generation function example.

The inter-route routing policy ρ_{MS} uses the set of route generation functions defined as $\Xi = \bigcup_{w \in \mathcal{W}, \tau \in T} \xi^{\tau}(w, C_k, i_k)$, where $T = \{RNN, RBI, RFI, RNI\}$. The cost

of each route in the assembler h_r is set to the total duration of the route T_r.

Potential-based routing policy

The routing policies described in Sections 4.4.3 and 4.4.3 ignore the arrival of new requests and instead focus on immediate benefits. As a result, they may yield suboptimal long-term decisions, where routes that seem efficient at first become inefficient over time. We propose a potential-based routing policy that builds on the multi-knapsack approximation model (4.12)-(4.15) and aims to remedy these issues. This policy takes into account the expected number of requests that can potentially be inserted into a given park-and-loop route. In particular, let us again consider the set of scenarios Σ. Let us also define the potential g_r of each route r as the expected number of on-demand requests that can be feasibly inserted. Then, the potential g_r of a route r can be computed by setting $\theta = \{r\}$ and solving formulation (4.12)-(4.15). This amounts to setting the routing plan to be equal to a single route r. Note, that in this case constraints (4.14) can be removed.

Using this definition, we define two potential-based routing policies that improve on policies ρ_{MI} and ρ_{MS}. The key difference with the myopic policies is that the cost of each route in the assembler h_r is set to the potential of the route g_r multiplied by -1. These two policies are denoted as ρ_{PI} and ρ_{PS}. Using these policies, one can expect the design of routing plans that would tend to better accommodate new on-demand requests. However, it must be noted that computing the potential of each route in the subset of routes generated by the route generation functions can be computationally expensive if the number of routes is too large.

4.4.4 Offline policies

A key step to solving the DPLRP is to build a high-quality initial routing plan at the initial state $\mathcal{S}_0$. In this section, we describe two policies that can be used to build such a plan.

Myopic offline routing policy

The first offline policy only considers the set of scheduled requests C^s. Let R be the set of all feasible park-and-loop routes that serve only the scheduled requests. Also, let a_{ir} be a parameter that takes the value 1 if route r serves request i, and 0 otherwise. Let c_r be the total distance covered using the vehicle in route r. Finally, let y_r be a binary variable that takes the value 1 if route r is selected as part of the initial routing plan and 0 otherwise. Then, an initial routing plan can be computed using the following set partitioning formulation:

$$\min \sum_{r \in R} c_r y_r \tag{4.18}$$

subject to

$$\sum_{r \in R} a_{ir} y_r = 1 \qquad \forall i \in C^s \tag{4.19}$$

$$\sum_{r \in R} y_r \leq |\mathcal{W}| \tag{4.20}$$

$$y_r \in \{0, 1\} \qquad \forall r \in R. \tag{4.21}$$

The objective function (4.18) minimizes the total distance covered by the vehicles. Constraints (4.19) guarantee that all scheduled requests are served. Constraints (4.20) impose a limit on the number of routes that are selected. Constraints (4.21) state the decision variables are binary. Generating the complete set of routes R may not be possible if the number of scheduled requests $|C^s|$ is large. Thus, exactly solving the mathematical formulation (4.18)-(4.21) is not always feasible in practice. Instead, one may rely on metaheuristics or column generation-based methods to generate a high-quality set of routes. In the context of the DPLRP, we define the route generation function $\xi^T(C^s)$ that builds a subset of routes $\overline{R}$ using as input the scheduled requests. This route generation function starts by building a TSP tour that includes all scheduled requests. Then, the TSP tour is partitioned into routes using the split algorithm proposed by Cabrera et al. (2022).

The myopic offline routing policy ϑ_M uses the set of route generation functions defined as $\Xi = \bigcup_{\tau \in T} \xi^{\tau}(C^s)$, where $T = \{RNN, RBI, RFI, RNI\}$. For the assembler that builds the initial routing plan, this policy uses formulation (4.18)-(4.21).

Potential-based offline routing policy

The previous policy may result in the assignment of very long routes to some workers, and very short ones to others. Some workers may even remain idle at the depot for some time, as the result of not being assigned to any of the routes. A shortcoming of this behavior is that in the long run, a higher number of on-demand requests will be rejected. Thus, in a similar fashion to the potential-based online routing policies, we use a set of scenarios Σ to compute the potential of each route g_r. To do so, we solve the multi-knapsack model (4.12)-(4.15) for each scenario in set Σ. Using the notation presented in the previous section, we can compute an initial routing plan with the following formulation:

$$\max \sum_{r \in R} g_r y_r \tag{4.22}$$

subject to (4.19)-(4.21).

The objective function (4.22) maximizes the total potential of the initial routing plan. The potential-based offline routing policy ϑ_P uses the set of route generation functions defined as $\Xi = \bigcup_{\tau \in T} \xi^{\tau}(C^s)$, where $T = \{RNN, RBI, RFI, RNI\}$. Again, the assembler that builds the initial routing plan solves formulation (4.19)-(4.22).

Table 4.2 summarizes our set of scheduling policies. Each policy is a combination of an offline policy (ϑ_M, ϑ_P), a routing policy ($\rho_{MS}, \rho_{MI}, \rho_{PS}, \rho_{PI}$), and an acceptance policy ($\varrho_G, \varrho_{MK-50}, \varrho_{R-50}$). We denote a scheduling policy by $\vartheta - \rho - \varrho$.

4.5 Perfect information bound

To further evaluate the quality of a policy, we establish a dual bound on the value of the optimal policy though the use of a perfect information relaxation. To compute

Table 4.2: Summary of the scheduling policies.

Scheduling policy	Offline policies		Routing policies				Acceptance policies		
	Myopic	Potential	Myopic best-insertion	Myopic Inter-route	Potential best-insertion	Potential Inter-route	Greedy	Potential	Rollout
$\vartheta_{M\text{-}\rho_{MI}\text{-}\varrho_G}$	X		X				X		
$\vartheta_{M\text{-}\rho_{MS}\text{-}\varrho_G}$	X			X			X		
$\vartheta_{M\text{-}\rho_{MI}\text{-}\varrho_{MK-50}}$	X		X					X	
$\vartheta_{M\text{-}\rho_{MS}\text{-}\varrho_{MK-50}}$	X			X				X	
$\vartheta_{M\text{-}\rho_{MI}\text{-}\varrho_{R-50}}$	X		X						X
$\vartheta_{M\text{-}\rho_{MS}\text{-}\varrho_{R-50}}$	X			X					X
$\vartheta_{P\text{-}\rho_{MI}\text{-}\varrho_G}$		X	X				X		
$\vartheta_{P\text{-}\rho_{MS}\text{-}\varrho_G}$		X		X			X		
$\vartheta_{P\text{-}\rho_{MI}\text{-}\varrho_{R-50}}$		X	X						X
$\vartheta_{P\text{-}\rho_{MS}\text{-}\varrho_{R-50}}$		X		X					X
$\vartheta_{P\text{-}\rho_{MS}\text{-}\varrho_{MK-50}}$		X		X				X	
$\vartheta_{P\text{-}\rho_{PI}\text{-}\varrho_G}$		X			X		X		
$\vartheta_{P\text{-}\rho_{PS}\text{-}\varrho_G}$		X				X	X		

this bound we must solve the DPLRP exactly under the assumption that all the stochastic information is known and is available before the start of the planning horizon. This means that all the arrival times of the on-demand requests are known. To mathematically state this problem we start by defining the set $\mathbf{\Omega}$ that contains all feasible park-and-loop routes. All the routes in this set satisfy the property that a request is never served before it is received. We also let a_{ir} be a binary parameter that takes the value 1 if route r serves request i, and 0 otherwise. Then, let y'_r be a binary variable that takes the value 1 if the park-and-loop route is selected, and 0 otherwise. Also, let z'_i be a binary variable that takes the value 1 if the on-demand request is served, and 0 otherwise. Then, the perfect information bound can be computed by solving the following mathematical integer program:

$$\max \sum_{i \in C^d} z'_i \tag{4.23}$$

subject to

$$\sum_{r \in \Omega} a_{ir} y'_r = 1 \qquad \forall i \in C^s \tag{4.24}$$

$$\sum_{r \in \Omega} a_{ir} y'_r = z'_i \qquad \forall i \in C^d \tag{4.25}$$

$$\sum_{r \in \Omega} y'_r \leq |\mathcal{W}| \tag{4.26}$$

$$y'_r \in \{0,1\} \qquad \forall r \in \Omega \tag{4.27}$$

$$z'_i \in \{0,1\} \qquad \forall i \in C^d. \tag{4.28}$$

The objective function (4.23) maximizes the number of on-demand requests served. Constraints (4.24) ensure that all scheduled requests are served. Constraints (4.25) state that on-demand requests can be served only once. They also guarantee that these requests can only be served by a route that visits their location. Constraints (4.26) impose an upper bound on the number of routes. Constraints (4.27)-(4.28) state that variables are binary. Formulation (4.23)-(4.28) assumes that the set of all park-and-loop routes Ω is known. However, the size $|\Omega|$ of the route set grows exponentially with the number of requests. As an alternative, one can resort to column generation-based approaches. In this work, we use, with mild changes, the branch-price-and-cut algorithm introduced in Cabrera et al. (2023). We specifically adjusted the pricing problem to incorporate time windows at customer locations and redefined the formula for calculating the reduced cost of a route.

4.6 Computational experiments

In this section, we present our computational experiments. Our goal is to analyze the performance of the different online and offline policies. All algorithms and simulations were implemented in Java and executed on an Intel core i7 @2.30 GHz Quad-Core processor with 12 GB of RAM. We used Gurobi version 9.5.1 to solve linear and integer programs. The branch-price-and-cut algorithm to obtain the PIB was implemented in Java using the jORLib[1] library. We rely on CPLEX 20.1 to solve the master problem. All instances, the corresponding solution files, and a solution checker are available at `https://github.com/ncabrera10/DPLRP-SC`.

[1]The latest version of jORLib can be downloaded at: http://coin-or.github.io/jorlib/.

4.6.1 Test instances

We created a new set of instances based on the testbed designed by Zhang et al. (2023). They constructed a graph comprising 16,080 nodes representing the city of Vienna, Austria, which is publicly available. Figure 4.4 illustrates the graph with a random sample of 150 nodes, where the depot location is indicated by a red icon. In the following sections, we first introduce the concept of the degree of dynamism, followed by a detailed description of the procedure used to generate each instance.

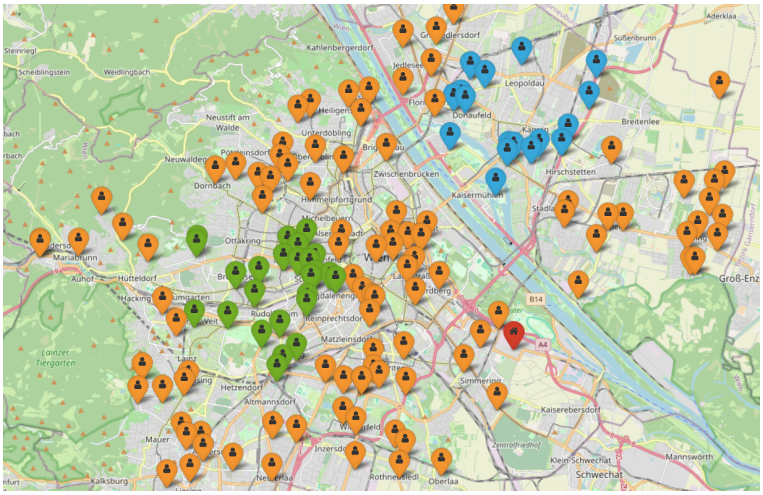


Figure 4.4: Sample of Vienna graph.

Degree of dynamism

The degree of dynamism is a measure of the speed and frequency with which new information is revealed in dynamic optimization problems. Originally introduced by Lund et al. (1996), the degree of dynamism δ is defined as the ratio of new requests to the total number of requests. Using our notation, δ is mathematically defined as:

$$\delta = \frac{|C^d|}{|C^d| + |C^s|} \in [0, 1] \,. \tag{4.29}$$

According to (4.29) the degree of dynamism increases if $|C^d|$ increases or if $|C^s| \geq 1$ decreases. When the degree of dynamism is low, the routing plan requires

infrequent updates, making the initial routing plan generated by offline policies particularly crucial. Conversely, when the degree of dynamism is high, the routing plan must be updated frequently, increasing the importance of effective acceptance and routing policies.

Instance design and parameters

We generated a total of 270 instances for this study. Each instance is denoted as $B\_\delta\_\lambda\_\Lambda\_n\_\mathcal{W}$, where $B = \{1, 2, 3, 4, 5\}$ identifies the random seed, δ is the degree of dynamism, λ is the arrival rate of on-demand requests, Λ corresponds to the spatiotemporal distribution of the on-demand requests, n is the total number of requests, and $\mathcal{W}$ is the number of workers. Table 4.3 details the parameters used. In Column 1, we list the number of requests, while Column 2 specifies the arrival rate in requests per minute. Column 3 indicates the degree of dynamism, and columns 4 and 5 show the number of on-demand and scheduled requests, respectively. In Column 6, we detail the number of workers. For each combination of parameters, we generated 15 instances. We calculated driving and walking times based on assumed speeds of 20 km/h and 4 km/h, respectively. The service time for each request ranges from 20 to 35 minutes, with a parking time set at 5 minutes. The maximum walking distance between two locations is 2.5 km, and each worker has a daily walking limit of 5 km. Each walking subtour lasts up to 2 hours, and the total planning horizon spans 7 hours. These parameters were selected based on a real-world application described in Cabrera et al. (2022).

The process to generate an instance is the following. First, we compute the number of on-demand requests as $\lceil f \times \lambda \rceil$. Then, we compute the number of scheduled requests $|\mathcal{C}^s|$ as $\left\lceil \frac{\mathcal{C}^d \times (1-\delta)}{\delta} \right\rceil$. Third, we sample $|\mathcal{C}^s|$ locations from the graph, which are assigned as scheduled requests. Then, we assign the arrival time of each on-demand request using a Poisson distribution with a mean equal to λ and we determine the location of each on-demand request. In this process, we consider three spatiotemporal distribution types: uniform time-independent (UTI), clustered time-independent

Table 4.3: Instances information.

n	λ	δ	$\lvert\mathcal{C}^d\rvert$	$\lvert\mathcal{C}^s\rvert$	$\lvert\mathcal{W}\rvert$
30	0.007	0.10	3	27	3
30	0.018	0.25	6	24	3
30	0.036	0.50	15	15	3
30	0.054	0.75	24	6	3
30	0.064	0.90	27	3	3
30	0.068	0.95	29	1	3
50	0.012	0.10	5	45	6
50	0.030	0.25	11	39	6
50	0.060	0.50	25	25	6
50	0.089	0.75	39	11	6
50	0.107	0.90	45	5	6
50	0.113	0.95	47	3	6
100	0.024	0.10	10	90	12
100	0.060	0.25	25	75	12
100	0.119	0.50	50	50	12
100	0.179	0.75	75	25	12
100	0.214	0.90	90	10	12
100	0.226	0.95	95	5	12

(CTI), and clustered time-dependent (CTD). For the UTI distribution, on-demand request locations are randomly sampled across the graph. In contrast, for the CTI distribution, requests are sampled with equal probability from two distinct clusters, as illustrated in Figure 4.4. These clusters, each with a 3 km radius, are centered at coordinates (48.2499, 16.4434) and (48.1924, 16.3158). Requests colored in blue and green belong to clusters one and two, respectively. Lastly, for the CTD distribution, the likelihood of selecting a node from the second cluster increases linearly over time, resulting in a higher concentration of on-demand requests in this cluster as the planning horizon approaches its end. Figure 4.5 depicts the request locations for instance 3_0.25_0.03_CTD_50_6.

4.6.2 Assessing the performance of the scheduling policies

In this section, we assess the performance of the scheduling policies. For all policies that use the MSH matheuristic, we set Q = 1000, Ψ = 60 seconds, and Φ = 10

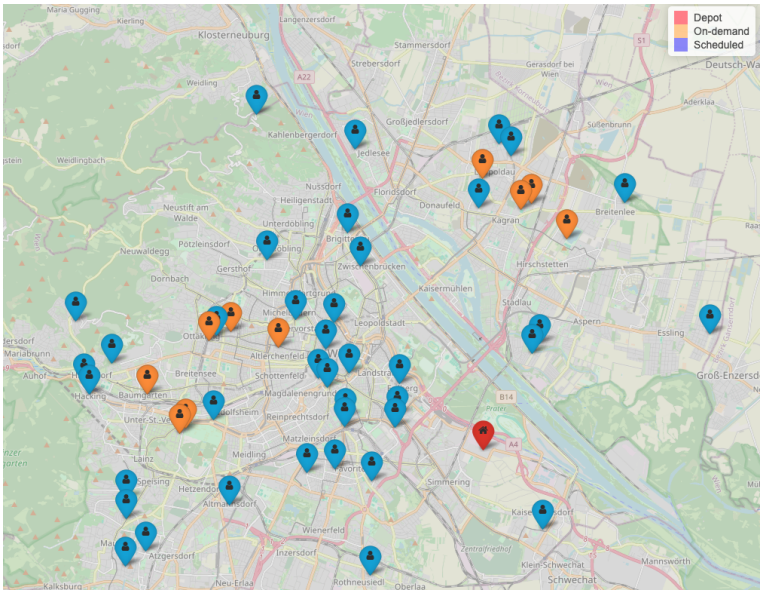


Figure 4.5: Instance 3_0.25_0.03_CTD_50_6.

seconds. To compute the perfect information bound we use the default parameters as described in Cabrera et al. (2023). To compare the scheduling policies we consider two metrics. First, the acceptance rate which is computed as the ratio between the number of on-demand requests served and the total number of on-demand requests received during the planning horizon. Second, the computational time required to select an action at each decision epoch.

The perfect information bound problem is at the intersection of two challenging combinatorial optimization problems: the team-orienteering problem and the park-and-loop routing problem with time windows. Hence, solving it can be a daunting task. Our preliminary experiments confirmed this intuition as the branch-price-and-cut algorithm was consistently able to solve all instances with 30 requests, but was only able to solve a few of the instances with 50 and 100 requests. For this reason, we first evaluate the scheduling policies in the subset of instances with 30 requests. Figure 4.6 compares the acceptance rate obtained using each scheduling policy considering this subset of instances. We depict the average perfect information bound with a red line. On average, the perfect information solutions serve 77.7% of the on-demand requests. As this figure shows, the $\vartheta_P - \rho_{MS} - \varrho_G$ and $\vartheta_P -$

$\rho_{MS} - \varrho_{R-50}$ scheduling policies are the ones that find the closes results. They fail to serve 20.07% of the maximum attainable demand (i.e., $\frac{77.7\%-62.1\%}{77.7\%} = 20.07\%$). This figure indicates that the scheduling policies that use the offline policy ϑ_P tend to outperform their counterparts that use the offline policy ϑ_M by increasing the acceptance rate up to 4 percentage points.

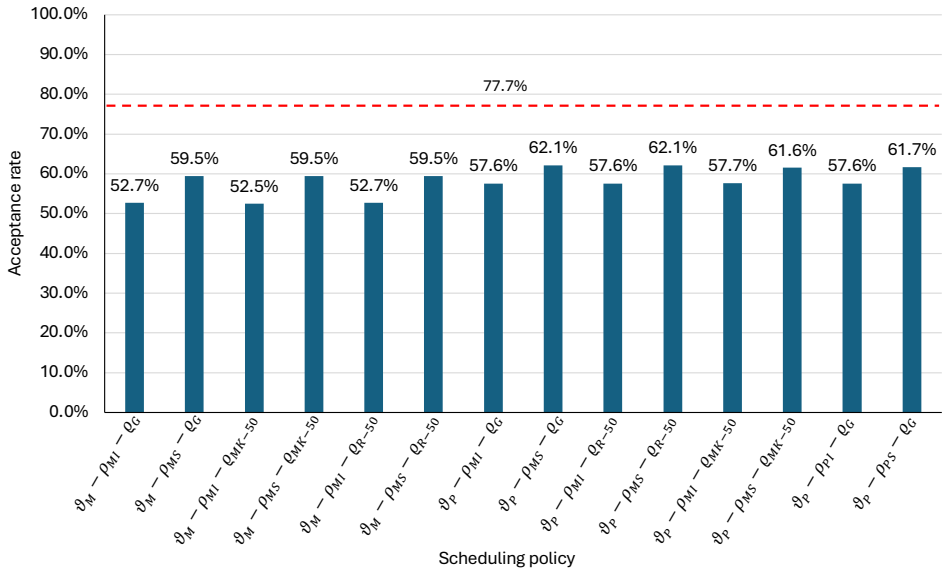


Figure 4.6: Comparison of scheduling policies acceptance rate and the perfect information bound in the subset of instances with 30 requests.

To further evaluate the performance of these scheduling policies, we also use the large instances with 50 and 100 requests. Figure 4.7 compares the performance of the proposed scheduling policies based on the average acceptance rate across these instances. The results show that the $\vartheta_P - \rho_{PS} - \varrho_G$ scheduling policy achieves the highest acceptance rate at 79.8%. Our results indicate that considering the uncertainty of new request arrivals both in the design of the initial routing plan and when updating the routing plan positively impacts the acceptance rate. In contrast, the $\vartheta_M - \rho_{MI} - \varrho_{MK-50}$ policy, which neither anticipates new request arrivals when building the initial routing plan nor allows for reassigning requests among workers during updates, exhibits the lowest acceptance rate. This lack of flexibility often leads to difficulties in adapting to new requests.

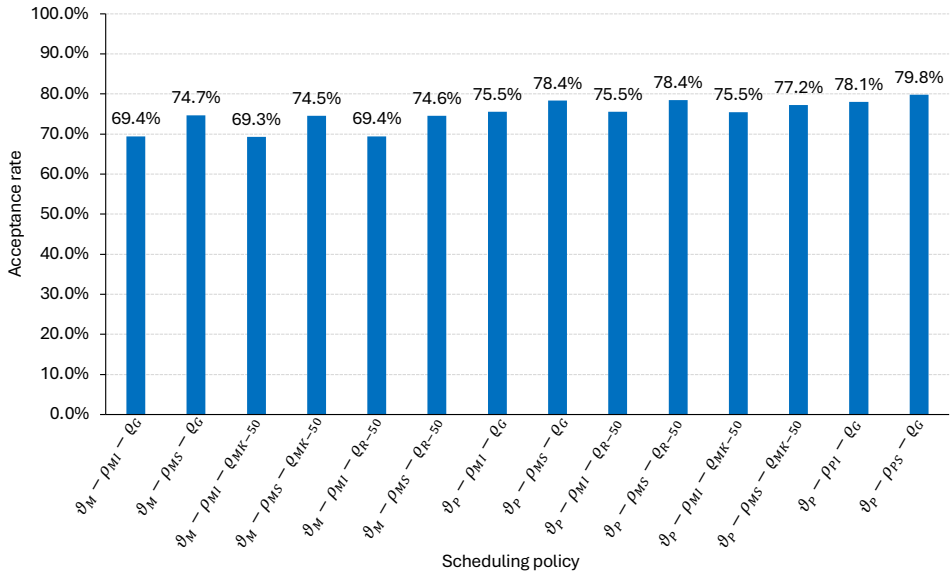


Figure 4.7: Comparison of scheduling policies acceptance rate in the subset of instances with 50 and 100 requests.

We also compare the scheduling policies based on the computational time required to make a decision when a new on-demand request appears. Figure 4.8 presents the average and maximum decision times, in seconds, across all instances. As shown, all policies make decisions in under one minute, regardless of the instance. However, scheduling policies $\vartheta_M - \rho_{MI} - \varrho_G$ and $\vartheta_P - \rho_{MI} - \varrho_G$ are the fastest. These two policies rely on the ρ_{PI} routing policy. In contrast, policies that use the ϱ_{R-50} acceptance policy tend to be slower on average, which is expected due to the high number of simulations required before making a decision.

4.6.3 Assessing the performance of the offline policies

In this section, we analyze the performance of the offline policies as the degree of dynamism varies in the set of instances with 50 and 100 requests. Figure 4.9 illustrates the average acceptance rate for each offline policy as a function of the degree of dynamism. The results indicate that policies using the potential-based offline policy consistently achieve a higher average acceptance rate, regardless of the degree of dynamism. However, the difference between using a myopic and an anticipatory

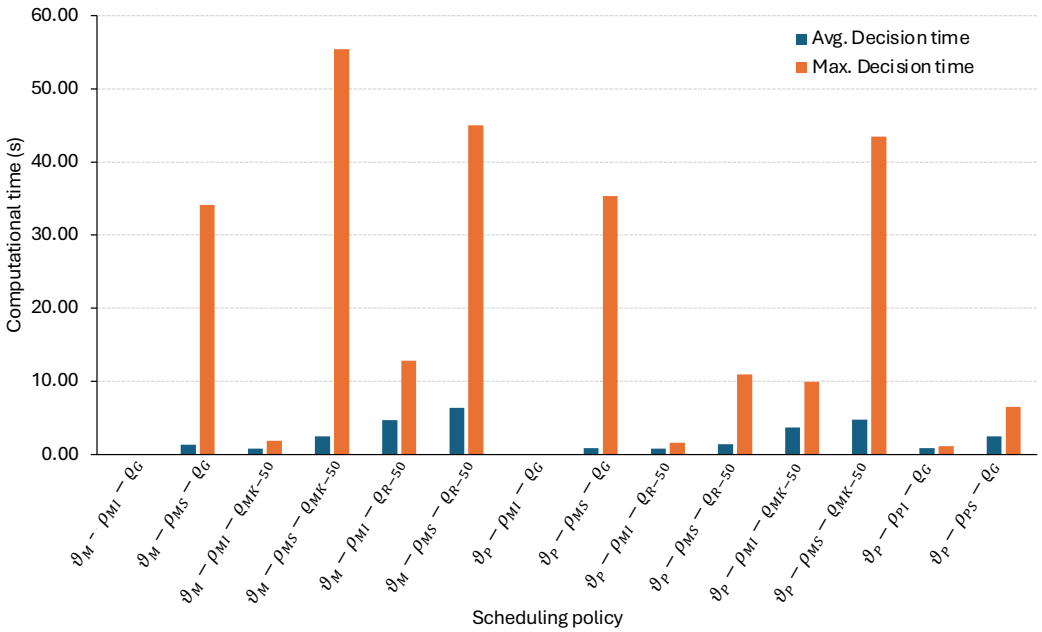


Figure 4.8: Comparison of scheduling policies computational effort in the subset of instances with 50 and 100 requests.

policy becomes more pronounced when the degree of dynamism approaches 0.5, where the expected number of on-demand requests roughly equals the number of scheduled requests. This suggests that the choice between myopic and anticipatory policies should be carefully considered, especially when operating in scenarios with a moderate degree of dynamism.

To further explain the performance differences between the offline policies, we begin by showcasing the routing plans generated by each approach on a given instance. Figures 4.10a and 4.10b depict the initial routing plans created by the myopic and potential-based offline policies for instance 1_0.5_0.06_CTD_50_6, which includes 27 scheduled requests and 23 on-demand requests, with six available workers. As these figures demonstrate, the initial routing plans from the two policies can differ significantly. The myopic policy, for instance, generates only three routes, leaving three workers idle at the depot. In contrast, the potential-based policy creates six routes, utilizing all available workers from the outset. Using these initial routing plans as inputs, we evaluated all scheduling policies and observed that the

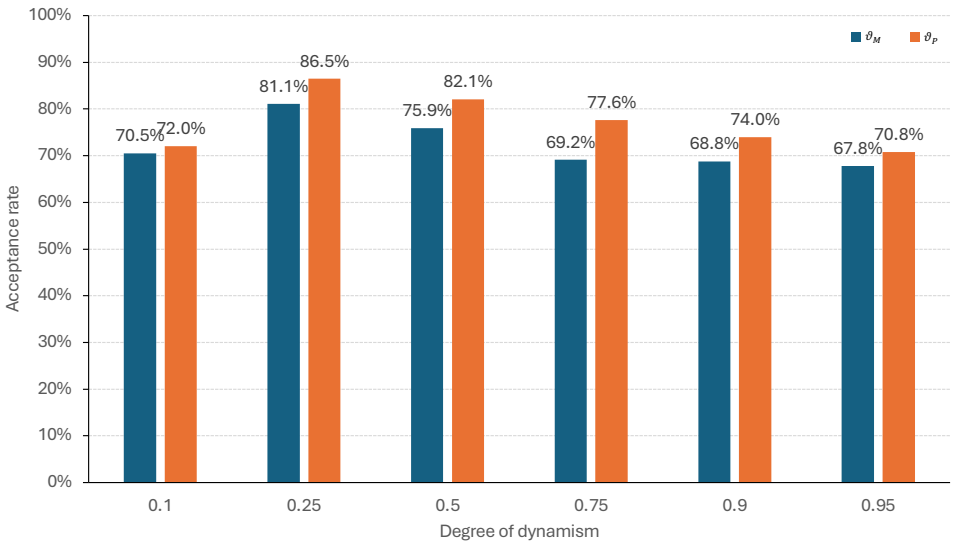
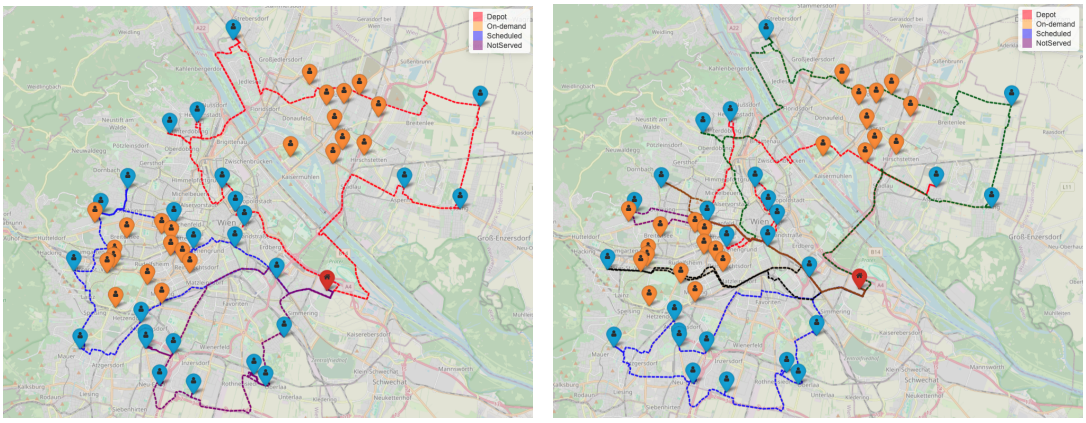


Figure 4.9: Comparison of the offline policies in the subset of instances with 50 and 100 requests.

maximum number of served on-demand requests was 17 for the myopic plan and 20 for the potential-based plan. We believe that engaging all workers from the start provides greater flexibility in handling on-demand requests.



(a) Myopic (b) Potential based

Figure 4.10: Initial routing plans on instance 1_0.5_0.06_CTD_50_6.

To further assess the impact of the offline policies, we compare the percentage of time that workers spend idle at the depot under each approach. Figure 4.11 shows the percentage of idle time as a function of the degree of dynamism. The results

indicate that when the degree of dynamism is low, idle times are minimal, nearing 0%. However, as the degree of dynamism increases, idle times can rise to 34.6% and 29.7%. The figure also demonstrates that, regardless of the degree of dynamism, the offline potential-based policy generally reduces idle time. In some cases, the difference between the two policies is as much as 11.2%, highlighting the efficiency gains from using the potential-based approach.

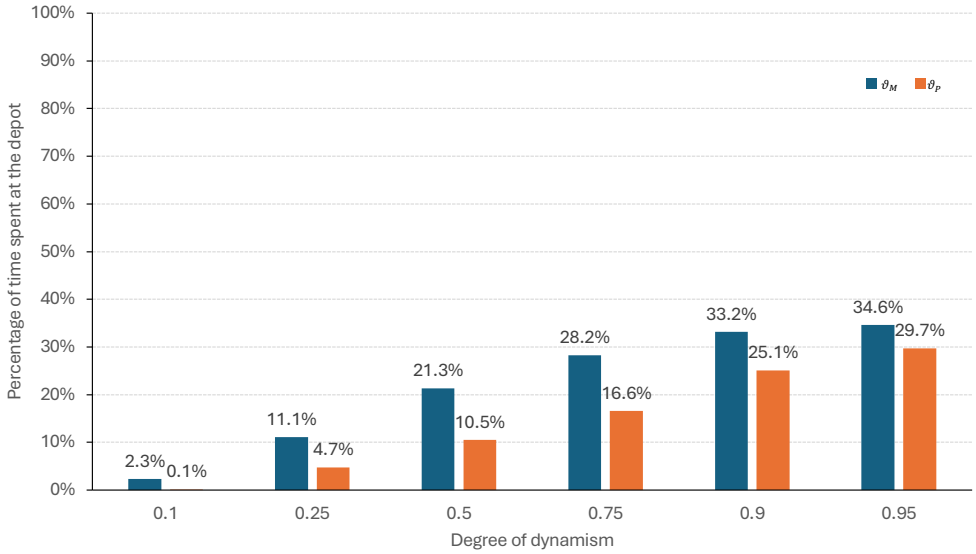


Figure 4.11: Comparison of the offline policies idle times in the subset of instances with 50 and 100 requests.

4.6.4 Assessing the performance of the online policies

In this section, we evaluate the performance of both the routing and the acceptance policies using the acceptance rate and the computational times in the set of instances with 50 and 100 requests. Figure 4.12 illustrates the average acceptance rate relative to the degree of dynamism. The results show that using routing policy ρ_{PS} generally improves the performance of the scheduling policies. Specifically, when comparing routing policy ρ_{MS} with routing policy ρ_{PS}, we find that the potential-based routing policy increases the acceptance rate by an average of 3.75%. This positive impact is even more pronounced when comparing routing policies ρ_{MI} and ρ_{PI}, where the

acceptance rate improves by an average of 5.55%. This increase can be attributed to the fact that these routing policies do not reassign requests between workers, making the anticipation of future requests more critical.

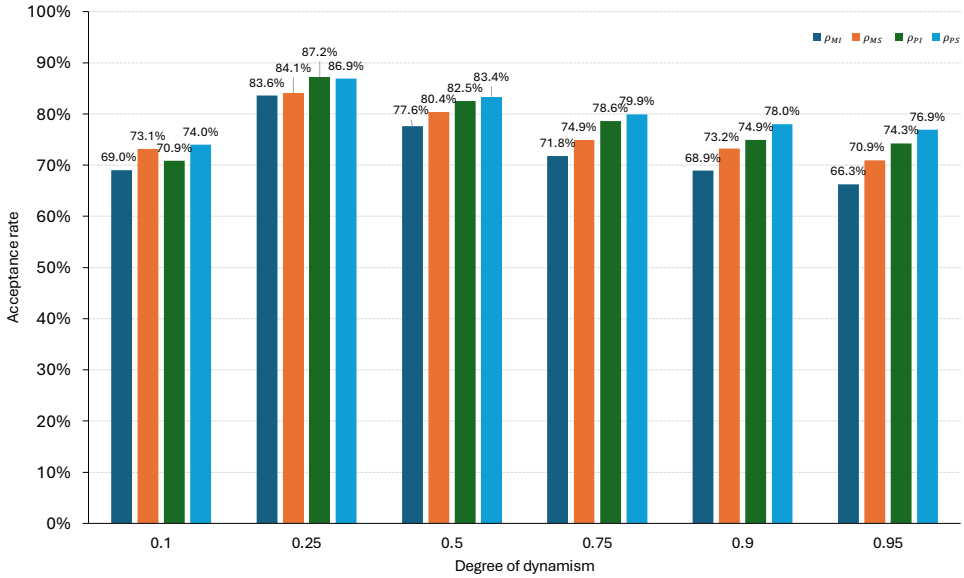


Figure 4.12: Comparison of the routing policies acceptance rate in the subset of instances with 50 and 100 requests.

We also compare the routing policies in terms of the computational time required to update the routing plan. Figure 4.13 shows both the average and maximum update times for each policy. As illustrated, routing policy ρ_{MI} consistently updates the routing plan in just a few milliseconds, making it the fastest on average, followed by routing policy ρ_{PI}. As mentioned earlier, these two policies do not reassign requests among workers, unlike policies ρ_{MS} and ρ_{PS}, which allows them to compute an updated routing plan much faster. However, note that routing policy ρ_{PI} has an average update time of nearly 0.9 seconds, indicating that calculating the potential for each route in the pool and solving the set partitioning problem with the updated objective is significantly more computationally demanding. A similar observation can be made when comparing routing policies ρ_{MS} and ρ_{PS}, where the average update time increases from 1.13 seconds to 2.51 seconds.

Concerning the acceptance policies, Figure 4.14 depicts the acceptance rate rela-

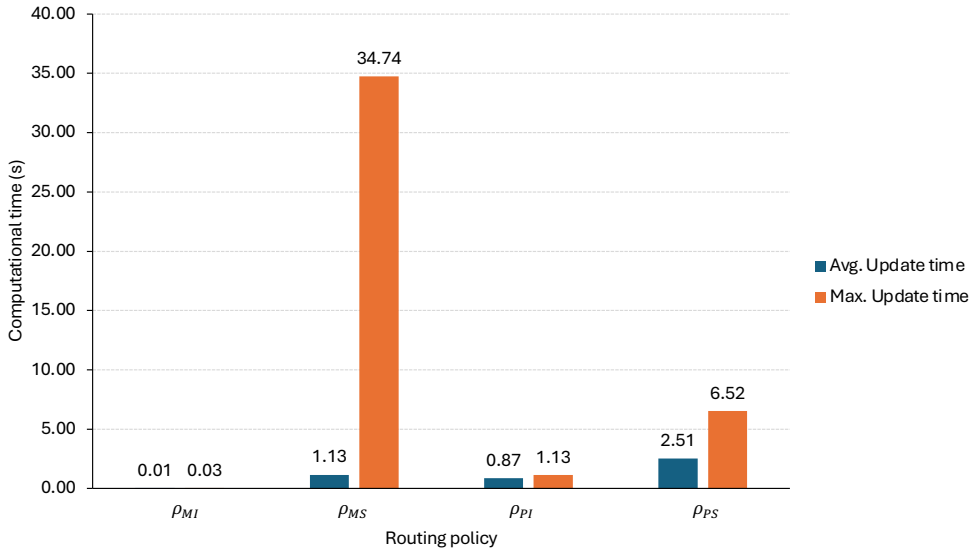


Figure 4.13: Comparison of the routing policies computational effort in the subset of instances with 50 and 100 requests.

tive to the degree of dynamism when considering instances with 50 and 100 requests. As this figure shows, using an anticipatory acceptance policy only marginally improves the performance of the scheduling policies. A possible explanation for this behavior is that the anticipatory policies ϱ_{R-50} and ϱ_{MK-50} may not be exploring a broad enough set of future possibilities. This limited exploration could lead to suboptimal decision-making, particularly in a problem as complex as the DPLRP, where small changes in decision variables can have significant impacts. Additionally, the greedy policy may be already performing near optimally, leaving little room for the rollout policy to offer substantial improvements.

We also compare the acceptance policies in terms of the computational time required to make an acceptance decision. Figure 4.15 shows the average and maximum decision times for each policy, when using the fastest routing policy, that is, ρ_{MI}. As this figure shows, the greedy policy ϱ_G can take a decision almost immediately. In contrast, the acceptance policies ϱ_{MK-50} and ϱ_{R-50} take almost 0.8 and 4.1 seconds on average to decide whether to accept or reject an on-demand request.

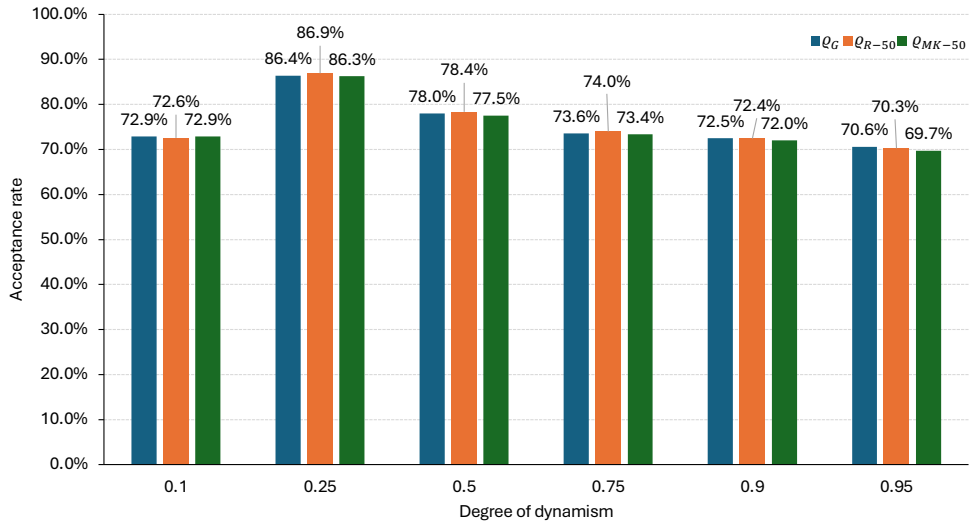


Figure 4.14: Comparison of the acceptance policies acceptance rates in the subset of instances with 50 and 100 requests.

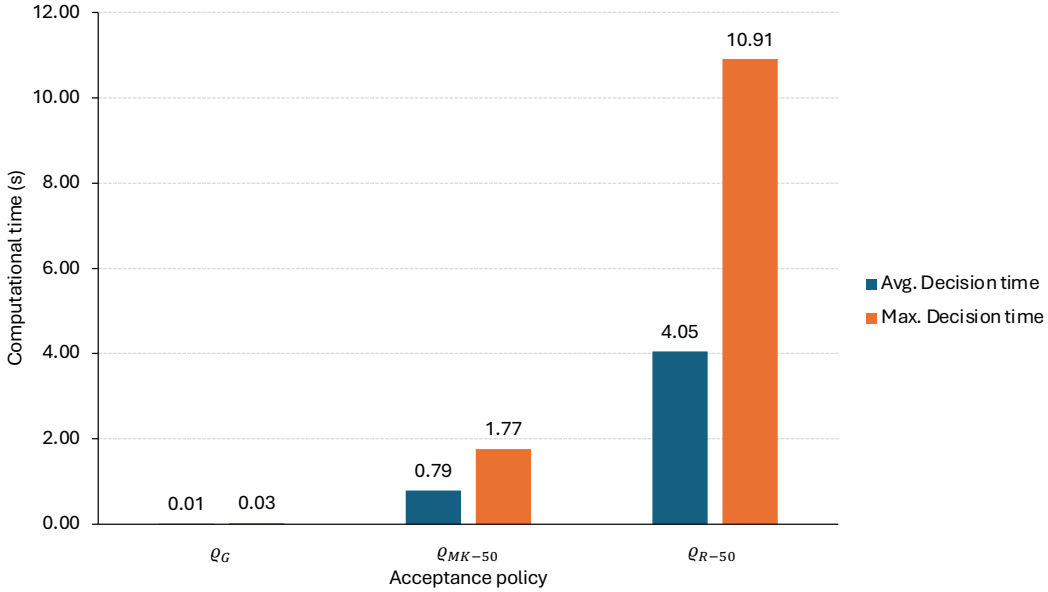


Figure 4.15: Comparison of the acceptance policies decision times in the subset of instances with 50 and 100 requests.

4.7 Concluding remarks

In this paper, we introduced the dynamic park-and-loop routing problem and proposed a route-based sequential decision process to address it. To solve this problem, we developed a set of scheduling policies that leverage the multi-space sampling

heuristic. The use of MSH enables real-time consideration of complex routing decisions, including extending or shortening existing walking subtours, inserting or removing subtours, reassigning requests between workers, and reordering requests within a route. Each scheduling policy integrates an offline routing policy, an acceptance policy, and a real-time routing policy, ensuring adaptability to dynamic conditions. Additionally, we formulated the perfect information bound problem, providing a benchmark for evaluating the effectiveness of our approach.

We evaluated our set of scheduling policies using a newly generated set of instances with up to 100 requests. Our computational results demonstrate that these policies can produce solutions that, on average, are within 15% of the perfect information bound, showcasing their effectiveness under dynamic conditions. Regarding offline routing policies, our experiments indicate that employing an anticipatory policy can significantly enhance decision-making by anticipating sources of uncertainty, leading to an improvement in the acceptance rate by up to 4.95% on average. Furthermore, our results highlight the superiority of the potential-based inter-route routing policy. This policy not only effectively anticipates the arrival of new requests but also facilitates the efficient reassignment of requests among workers, ensuring a more responsive routing plan. These findings underscore the importance of incorporating anticipation mechanisms in both offline and real-time routing decisions to improve overall system performance.

Future research should focus on incorporating other sources of uncertainty into the DPLRP such as stochastic travel times between locations and stochastic service times. These factors could significantly impact the performance of the proposed routing policies, and accounting for them would enhance the robustness and adaptability of the solutions in real-world scenarios.

References

Belenguer, J., E. Benavent, A. Martinez, C. Prins, C. Prodhon, and J. Villegas (2016). "A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots". *Transportation Science* 50, pp. 735–749.

Bent, R. and P. Van Hentenryck (Jan. 2007). "Waiting and relocation strategies in online stochastic vehicle routing". *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI-07)*. Hyderabad, India, pp. 1816–1821.

Bent, R. W. and P. Van Hentenryck (2004). "Scenario-based planning for partially dynamic vehicle routing with stochastic customers". *Operations Research* 52.6, pp. 977–987.

Bertsekas, D. P., J. N. Tsitsiklis, and C. Wu (1997). "Rollout algorithms for combinatorial optimization". *Journal of Heuristics* 3, pp. 245–262.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2025). "The workforce scheduling and routing problem with park-and-loop". *Networks* 85.1, pp. 38–60.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2022). "The doubly open park-and-loop routing problem". *Computers & Operations Research* 143, p. 105761.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2023). "Solving the park-and-loop routing problem by branch-price-and-cut". *Transportation Research Part C: Emerging Technologies* 157, p. 104369.

Chao, I.-M. (2002). "A tabu search method for the truck and trailer routing problem". *Computers and Operations Research* 29, pp. 33–51.

Chen, Z.-L. and H. Xu (2006). "Dynamic column generation for dynamic vehicle routing with time windows". *Transportation Science* 40.1, pp. 74–88.

Coindreau, M.-A., O. Gallay, and N. Zufferey (2019). "Vehicle routing with transportable resources: Using carpooling and walking for on-site services". *European Journal of Operational Research* 279, pp. 996–1010.

Derigs, U., M. Pullmann, and U. Vogel (2013). "Truck and trailer routing - Problems, heuristics and computational experience". *Computers and Operations Research* 40, pp. 536–546.

Echeverri, L. C., A. Froger, J. E. Mendoza, and E. Néron (July 2019). "A matheuristic for the Multi-period Electric Vehicle Routing Problem". *13th Metaheuristics International Conference*. Cartagena de Indias, Colombia.

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

Fontecha, J., R. Akhavan-Tabatabaei, D. Duque, A. Medaglia, M. Torres, and J. Rodríguez (2016). "On the preventive management of sediment-related sewer blockages: a combined maintenance and routing optimization approach". *Water Science & Technology* 20, pp. 302–308.

Gendreau, M., F. Guertin, J.-Y. Potvin, and É. Taillard (1999). "Parallel tabu search for real-time vehicle routing and dispatching". *Transportation Science* 33.4, pp. 381–390.

Gómez, A., R. Mariño, R. Akhavan-Tabatabaei, A. Medaglia, and J. Mendoza (2015). "On modeling stochastic travel and service times in vehicle routing". *Transportation Science* 50, pp. 627–614.

Le Colleter, T., D. Dumez, F. Lehuédé, and O. Péton (2023). "Small and large neighborhood search for the park-and-loop routing problem with parking selection". *European Journal of Operational Research* 308.3, pp. 1233–1248.

Lund, K., O. B. Madsen, and J. M. Rygaard (1996). *Vehicle routing problems with varying degrees of dynamism*. IMM, Institute of Mathematical Modelling, Technical University of Denmark.

Maghfiroh, M. F. and S. Hanaoka (2018). "Dynamic truck and trailer routing problem for last mile distribution in disaster response". *Journal of Humanitarian Logistics and Supply Chain Management* 8.2, pp. 252–278.

Martinez-Sykora, A., F. McLeod, C. Lamas-Fernandez, T. Bektaş, T. Cherrett, and J. Allen (2020). "Optimised solutions to the last-mile delivery problem in London using a combination of walking and driving". *Annals of Operations Research* 295.2, pp. 645–693.

Mendoza, J. E. and J. G. Villegas (2013). "A multi-space sampling heuristic for the vehicle routing problem with stochastic demands". *Optimization Letters* 7, pp. 1503–1516.

Mitrović-Minić, S. and G. Laporte (2004). "Waiting strategies for the dynamic pickup and delivery problem with time windows". *Transportation Research Part B: Methodological* 38.7, pp. 635–655.

Montoya, A., C. Guéret, J. E. Mendoza, and J. G. Villegas (2016). "A multi-space sampling heuristic for the green vehicle routing problem". *Transportation Research Part C: Emerging Technologies* 70, pp. 113–128.

Parragh, S. N. and J.-F. Cordeau (2017). "Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows". *Computers and Operations Research* 83, pp. 28–44.

Pillac, V., C. Guéret, and A.-L. Medaglia (2018). "A fast reoptimization approach for the dynamic technician routing and scheduling problem". *Recent Developments in Metaheuristics*, pp. 347–367.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons.

Rothenbächer, A.-K., M. Drexl, and S. Irnich (2018). "Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows". *Transportation Science* 52.5, pp. 1174–1190.

Sarasola, B., K. F. Doerner, V. Schmid, and E. Alba (2016). "Variable neighborhood search for the stochastic and dynamic vehicle routing problem". *Annals of Operations Research* 236, pp. 425–461.

Secomandi, N. (2001). "A rollout policy for the vehicle routing problem with stochastic demands". *Operations Research* 49.5, pp. 796–802.

Semet, F. (1995). "A two-phase algorithm for the partial accessibility constrained vehicle routing problem". *Annals of Operations Research* 61, pp. 45–65.

Soeffker, N., M. W. Ulmer, and D. C. Mattfeld (2024). "Balancing resources for dynamic vehicle routing with stochastic customer requests". *OR Spectrum*, pp. 1–43.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and M. Hennig (2019). "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests". *Transportation Science* 53.1, pp. 185–202.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and B. W. Thomas (2020). "On modeling stochastic dynamic vehicle routing problems". *EURO Journal on Transportation and Logistics* 9.2, p. 100008.

Ulmer, M. W., D. C. Mattfeld, and F. Köster (2018). "Budgeting time for dynamic vehicle routing with stochastic customer requests". *Transportation Science* 52.1, pp. 20–37.

Ulmer, M. W. and S. Streng (2019). "Same-day delivery with pickup stations and autonomous vehicles". *Computers & Operations Research* 108, pp. 1–19.

Ulmer, M. W. and B. W. Thomas (2018). "Same-day delivery with heterogeneous fleets of drones and vehicles". *Networks* 72.4, pp. 475–505.

Ulmer, M. W. and B. W. Thomas (2020). "Meso-parametric value function approximation for dynamic customer acceptances in delivery routing". *European Journal of Operational Research* 285.1, pp. 183–195.

Ulmer, M. W. (2017). *Approximate dynamic programming for dynamic vehicle routing*. Vol. 61. Springer.

Villegas, J., C. Prins, A. Medaglia, and N. Velasco (2013). "A matheuristic for the truck and trailer routing problem". *European Journal of Operational Research* 230, pp. 231–244.

Voccia, S. A., A. M. Campbell, and B. W. Thomas (2019). "The same-day delivery problem for online purchases". *Transportation Science* 53.1, pp. 167–184.

Zhang, J., K. Luo, A. M. Florio, and T. Van Woensel (2023). "Solving large-scale dynamic vehicle routing problems with stochastic requests". *European Journal of Operational Research* 306.2, pp. 596–614.

Zhang, J. and T. Van Woensel (2023). "Dynamic vehicle routing with random requests: A literature review". *International Journal of Production Economics* 256, p. 108751.

Zhou, H., H. Qin, C. Cheng, and L.-M. Rousseau (2023). "An exact algorithm for the two-echelon vehicle routing problem with drones". *Transportation Research Part B: Methodological* 168, pp. 124–150.

General Conclusion

This thesis offers an in-depth analysis of the workforce scheduling and routing problem, which involves assigning geographically dispersed tasks to workers and designing cost-effective routes to minimize the total expected cost. Within this context, we explored three extensions of the problem, with a particular emphasis on multi-modal routes. To address these extensions, we developed and evaluated various solution methodologies. The remainder of this section provides an overview of the chapters in this thesis, followed by a detailed discussion of its key contributions. Finally, we outline potential directions for future research.

In Chapter 1, we presented a comprehensive review of the workforce scheduling and routing problem. In our review, we described the key features previously addressed in the literature and identified promising research areas. We also introduced two mathematical formulations for the problem and presented an overview of the solution methods that have been used to tackle this family of problems. Finally, we discussed multiple features to build more applicable and successful solutions.

In Chapter 2, we introduced the park-and-loop routing problem and described a branch-price-and-cut algorithm capable of optimally solving small and medium size instances. We also introduced and solved an extension of the elementary shortest path problem with resource constraints that allows for the design of multi-modal routes. Finally, we investigated the impact of introducing park-and-loop routes in both the solution structure and the total distance covered using a vehicle.

In Chapter 3, we focused on the workforce scheduling and routing problem with

park-and-loop. We addressed a general setting in which workers may have different skills and in which tasks may require one or more skills, each at a potentially different level of proficiency. We described a general branch-price-and-cut algorithm and two mathematical formulations. Finally, we showed how the proposed methods could also be used to solve the technician routing and scheduling problem.

In Chapter 4, we studied the dynamic park-and-loop routing problem that extends the park-and-loop routing problem by considering the arrival of on-demand requests. We presented a set of myopic and anticipatory scheduling policies based on the multi-space sampling heuristic. Also, we investigated the impact of using different methods to build the initial routing plan. Finally, we showed the value of allowing the reassignment of requests between workers in a dynamic setting.

Contributions

This thesis contributes to the literature in several ways. First, it introduces three new problems together with sophisticated solution methods to solve them: the park-and-loop routing problem (PLRP), the workforce scheduling and routing problem with park-and-loop (WSRP-PL), and the dynamic park-and-loop routing problem (DPLRP). Second, this thesis extends the pulse algorithm for solving variants of the well-known elementary shortest path problem with resource constraints. Finally, this thesis provides many test instances for these new problems as well as online tools to visualize and check new solutions. The proposed methods are evaluated through extensive computational experiments in terms of their solution quality and efficiency.

I. Improving and Extending the Pulse Algorithm

Typically, column generation-based algorithms for solving vehicle routing problems require the solution of an elementary shortest path problem with resource con-

straints. In this work, we extended the pulse algorithm for allowing the design of park-and-loop routes and the selection of workers based on their skills. We presented a set of new pruning strategies that crucially improve the performance of the algorithm. We also compared the pulse algorithm with a traditional labeling algorithm under different parameters settings.

II. A Branch-price-and-cut Algorithm for the PLRP and the WSRP-PL

We proposed two branch-price-and-cut algorithms for solving the PLRP and the WSRP-PL. Our algorithms leverage on state-of-the-art techniques for initializing the set of columns, solving the pricing problem, and strengthening the relaxation of the set partitioning formulation via valid inequalities. We compared our algorithms with benchmark algorithms from the literature and found encouraging results. In particular, we found 11 previously unknown optimal solutions for the PLRP. We solved 241 out of 324 of the WSRP-PL instances to optimality and we obtained nine new best known solutions for the technician routing and scheduling problem (TRSP) instances. Finally, we extended the branch-price-and-cut algorithm proposed in Chapter 2, to obtain a perfect information bound for the DPLRP instances.

III. Value of Multi-modal Routes

Most of the research on the workforce scheduling and routing problem considers a single transportation mode. We addressed this gap in the literature by investigating the value of designing multi-modal routes consisting of a main tour that is completed using a vehicle (i.e., a car, a van, a truck) and subtours, that are carried out on foot after safely parking the vehicle. We developed new algorithms that are able to provide optimal park-and-loop routes for many instances in the literature. Moreover, through extensive numerical experiments we showed that multi-modal routes can

decrease the average driven distance up to 18.4% on average without increasing the size of the fleet.

IV. Development of Test Data Sets And Visualization Tools

We created 40 large instances for the PLRP (Chapter 2). These instances comprise between 60 to 90 customers. We also provided a lower bound on the objective function for each of these instances. For the WSRP-PL we created a set of 324 instances (Chapter 3) that we derived from a classic testbed from the literature. These instances, with both tight and loose time windows, consider between 25 to 75 customers. Finally, for the DPLRP (Chapter 4) we created 270 instances with 30, 50, and 100 requests using data from the city of Vienna. All instances are available at `http://chairelogistique.hec.ca/en/scientific-data/`. An online solution checker in which researchers can upload their own solutions for plotting and checking is also available.

V. Extensive Computational Experiments

We report the results of extensive computational experiments on all generated instances for the PLRP, the WSRP-PL, and the DPLRP. We also report results on the stantard data sets for the PLRP and the TRSP proposed by Kovacs et al. (2012). Overall we consider 728 instances. The results on the PLRP instances indicated that our branch-price-and-cut outperforms the state-of-the-art heuristics in terms of solution quality without a significant increase in computing time. The results on the WSRP-PL and the TRSP showed that our method performs well on instances with tight time windows and demonstrate the effectiveness of the proposed pruning strategies.

Future Work

This thesis opens several paths for future research. The possibilities can be categorized in two categories, modeling, and solution methodology. In what follows, we describe the ideas for each category.

I. Modeling Perspective

One of the challenges faced by the utilities is the location of the customers. Indeed, in most cases, customers are located in highly populated areas. Vehicles transiting in these areas are subject to traffic delays, mixed traffic flow, collisions, or even changing weather conditions. In contrast, pedestrians transit mostly without unexpected interruptions. This highlights an opportunity to extend the park-and-loop routing problem to consider stochastic driving travel times between locations while still considering deterministic walking travel times.

Another interesting path for future research is to extend the workforce scheduling and routing problem with park-and-loop by adding the possibility of carpooling. The possibility of sharing a vehicle between workers opens a wide range of options to improve efficiency and increase the amount of customers than can be served within a given workday. Exploring methods and algorithms to handle the synchronization between drivers and passengers can be very engaging.

II. Solution Methodology

Solving the elementary shortest path problem with resource constraints and park-and-loop efficiently is a critical step of solving park-and-loop routing problems. Adapting and applying the multi-space sampling heuristic to solve this problem during the first column generation iterations is a promising research avenue. Alter-

natively, combining the pulse algorithm and the ng-paths relaxation proposed by Baldacci et al. (2011) can certainly be an interesting path for further research.

The branch-price-and-cut algorithm presented in Chapter 2 has the potential to be applied to many related problems. In particular, adapting this method to solve other multi-modal optimization problems such as the truck-and-drone routing problem and the truck-and-trailer routing problem is another interesting path for new research.

Bibliography

Akjiratikarl, C., P. Yenradee, and P. R. Drake (2007). "PSO-based algorithm for home care worker scheduling in the UK". *Computers & Industrial Engineering* 53.4, pp. 559–583.

Algethami, H., A. Martínez-Gavara, and D. Landa-Silva (2019). "Adaptive multiple crossover genetic algorithm to solve workforce scheduling and routing problem". *Journal of Heuristics* 25, pp. 753–792.

Anoshkina, Y. and F. Meisel (2019). "Technician teaming and routing with service-, cost-and fairness-objectives". *Computers & Industrial Engineering* 135, pp. 868–880.

Arslan, O., O. Jabali, and G. Laporte (2018). "Exact solution of the evasive flow capturing problem". *Operations Research* 66.6, pp. 1625–1640.

Baldacci, R., A. Mingozzi, and R. Roberti (2011). "New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem". *Operations Research* 59.5, pp. 1269–1283.

Bard, J. F., Y. Shao, and A. I. Jarrah (2014). "A sequential GRASP for the therapist routing and scheduling problem". *Journal of Scheduling* 17, pp. 109–133.

Bazirha, M., R. Benmansour, and A. Kadrani (2023). "An efficient two-phase heuristic for the home care routing and scheduling problem". *Computers & Industrial Engineering* 181, p. 109329.

Belenguer, J., E. Benavent, A. Martinez, C. Prins, C. Prodhon, and J. Villegas (2016). "A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots". *Transportation Science* 50, pp. 735–749.

Bent, R. and P. Van Hentenryck (Jan. 2007). "Waiting and relocation strategies in online stochastic vehicle routing". *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI-07)*. Hyderabad, India, pp. 1816–1821.

Bent, R. W. and P. Van Hentenryck (2004). "Scenario-based planning for partially dynamic vehicle routing with stochastic customers". *Operations Research* 52.6, pp. 977–987.

Bertsekas, D. P., J. N. Tsitsiklis, and C. Wu (1997). "Rollout algorithms for combinatorial optimization". *Journal of Heuristics* 3, pp. 245–262.

Bolívar, M. A., L. Lozano, and A. L. Medaglia (2014). "Acceleration strategies for the weight constrained shortest path problem with replenishment". *Optimization Letters* 8.8, pp. 2155–2172.

Braekers, K., R. F. Hartl, S. N. Parragh, and F. Tricoire (2016). "A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience". *European Journal of Operational Research* 248.2, pp. 428–443.

Bredström, D. and M. Rönnqvist (2008). "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints". *European Journal of Operational Research* 191.1, pp. 19–31.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2025). "The workforce scheduling and routing problem with park-and-loop". *Networks* 85.1, pp. 38–60.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2022). "The doubly open park-and-loop routing problem". *Computers & Operations Research* 143, p. 105761.

Cabrera, N., J.-F. Cordeau, and J. E. Mendoza (2023). "Solving the park-and-loop routing problem by branch-price-and-cut". *Transportation Research Part C: Emerging Technologies* 157, p. 104369.

Cabrera, N., A. L. Medaglia, L. Lozano, and D. Duque (2020). "An exact bidirectional pulse algorithm for the constrained shortest path". *Networks* 76.2, pp. 128–146.

Çakırgil, S., E. Yücel, and G. Kuyzu (2020). "An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems". *Computers & Operations Research* 118, p. 104908.

Castillo-Salazar, J. A., D. Landa-Silva, and R. Qu (2016). "Workforce scheduling and routing problems: literature survey and computational study". *Annals of Operations Research* 239, pp. 39–67.

Chao, I.-M. (2002). "A tabu search method for the truck and trailer routing problem". *Computers and Operations Research* 29, pp. 33–51.

Chen, X., B. W. Thomas, and M. Hewitt (2016). "The technician routing problem with experience-based service times". *Omega* 61, pp. 49–61.

Chen, Z.-L. and H. Xu (2006). "Dynamic column generation for dynamic vehicle routing with time windows". *Transportation Science* 40.1, pp. 74–88.

Clapper, Y., J. Berkhout, R. Bekker, and D. Moeke (2023). "A model-based evolutionary algorithm for home health care scheduling". *Computers & Operations Research* 150, p. 106081.

Coindreau, M.-A., O. Gallay, and N. Zufferey (2019). "Vehicle routing with transportable resources: Using carpooling and walking for on-site services". *European Journal of Operational Research* 279, pp. 996–1010.

Contardo, C. and R. Martinelli (2014). "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". *Discrete Optimization* 12, pp. 129–146.

Cordeau, J.-F., G. Laporte, F. Pasin, and S. Ropke (2010). "Scheduling technicians and tasks in a telecommunications company". *Journal of Scheduling* 13, pp. 393–409.

Corredor-Montenegro, D., N. Cabrera, R. Akhavan-Tabatabaei, and A. L. Medaglia (2021). "On the shortest α-reliable path problem". *Top* 29.1, pp. 287–318.

Costa, L., C. Contardo, and G. Desaulniers (2019a). "Exact Branch-Price-and-Cut Algorithms for Vehicle Routing". *Transportation Science* 53, pp. 946–985.

Costa, L., C. Contardo, and G. Desaulniers (2019b). "Exact branch-price-and-cut algorithms for vehicle routing". *Transportation Science* 53.4, pp. 946–985.

Delavernhe, F., B. Castanier, C. Guéret, and J. E. Mendoza (2024). "The joint maintenance operation selection and technician routing problem". *Computers & Operations Research* 167, p. 106667.

Dellaert, N., F. Dashty Saridarq, T. Van Woensel, and T. G. Crainic (2019). "Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows". *Transportation Science* 53.2, pp. 463–479.

Derigs, U., M. Pullmann, and U. Vogel (2013). "Truck and trailer routing - Problems, heuristics and computational experience". *Computers and Operations Research* 40, pp. 536–546.

Desrosiers, J., M. Lübbecke, G. Desaulniers, and J. B. Gauthier (2024). *Branch-and-Price*. Les Cahiers du GERAD G-2024-36. GERAD, Montréal QC H3T 2A7, Canada: Groupe d'études et de recherche en analyse des décisions, pp. 1–657.

Dohn, A., E. Kolind, and J. Clausen (2009). "The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach". *Computers & Operations Research* 36.4, pp. 1145–1157.

Dohn, A. and M. Sevel (2008). "The home care crew scheduling problem". *Conference Proceedings of the 1st International Conference on Applied Operational Research*.

Du, J. and X. Wang (2024). "A branch-and-price-and-cut algorithm for the home health care routing and scheduling problem with multiple prioritized time windows". *Computers & Operations Research* 170, p. 106749.

Duque, D., L. Lozano, and A. L. Medaglia (2014). "Solving the orienteering problem with time windows via the pulse framework". *Computers and Operations Research* 54, pp. 168–176.

Duque, D., L. Lozano, and A. L. Medaglia (2015). "An exact method for the biobjective shortest path problem for large-scale road networks". *European Journal of Operational Research* 242.3, pp. 788–797.

Echeverri, L. C., A. Froger, J. E. Mendoza, and E. Néron (July 2019). "A matheuristic for the Multi-period Electric Vehicle Routing Problem". *13th Metaheuristics International Conference*. Cartagena de Indias, Colombia.

Estellon, B., F. Gardi, and K. Nouioua (2009). "High-performance local search for task scheduling with human resource allocation." *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics* 5752, pp. 1–16.

Facchinetti, G., D. D'Angelo, M. Piredda, T. Petitti, M. Matarese, A. Oliveti, and M. G. De Marinis (2020). "Continuity of care interventions for preventing hospital readmission of older people with chronic diseases: A meta-analysis". *International Journal of Nursing Studies* 101, p. 103396.

Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen (2004). "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". *Networks* 44.3, pp. 216–229.

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

Fırat, M. and C. A. Hurkens (2012). "An improved MIP-based approach for a multi-skill workforce scheduling problem". *Journal of Scheduling* 15.3, pp. 363–380.

Fontecha, J., R. Akhavan-Tabatabaei, D. Duque, A. Medaglia, M. Torres, and J. Rodríguez (2016). "On the preventive management of sediment-related sewer blockages: a combined maintenance and routing optimization approach". *Water Science & Technology* 20, pp. 302–308.

Gastaldi, M., R. Rossi, and G. Gecchele (2014). "Effects of driver task-related fatigue on driving performance". *Procedia-Social and Behavioral Sciences* 111, pp. 955–964.

Gendreau, M., F. Guertin, J.-Y. Potvin, and É. Taillard (1999). "Parallel tabu search for real-time vehicle routing and dispatching". *Transportation Science* 33.4, pp. 381–390.

Gjevjon, E. R., T. I. Romøren, B. Ø. Kjøs, and R. Hellesø (2013). "Continuity of care in home health-care practice: two management paradoxes". *Journal of Nursing Management* 21.1, pp. 182–190.

Goel, A. and F. Meisel (2013). "Workforce routing and scheduling for electricity network maintenance with downtime minimization". *European Journal of Operational Research* 231.1, pp. 210–228.

Gómez, A., R. Mariño, R. Akhavan-Tabatabaei, A. Medaglia, and J. Mendoza (2015). "On modeling stochastic travel and service times in vehicle routing". *Transportation Science* 50, pp. 627–614.

Grenouilleau, F., N. Lahrichi, and L.-M. Rousseau (2020). "New decomposition methods for home care scheduling with predefined visits". *Computers & Operations Research* 115, p. 104855.

Grenouilleau, F., A. Legrain, N. Lahrichi, and L.-M. Rousseau (2019). "A set partitioning heuristic for the home health care routing and scheduling problem". *European Journal of Operational Research* 275.1, pp. 295–303.

Gu, H., Y. Zhang, and Y. Zinder (2022). "An efficient optimisation procedure for the workforce scheduling and routing problem: Lagrangian relaxation and iterated local search". *Computers & Operations Research* 144, p. 105829.

Guastaroba, G., J.-F. Côté, and L. C. Coelho (2021). "The multi-period workforce scheduling and routing problem". *Omega* 102, p. 102302.

Günther, M. and V. Nissen (2013). "Application of Particle Swarm Optimization to the British Telecom Workforce Scheduling Problem". *Proceedings of the 9th international conference on the practice and theory of automated timetabling (PATAT 2012)*, pp. 242–256.

Hashimoto, H., S. Boussier, M. Vasquez, and C. Wilbaut (2011). "A GRASP-based approach for technicians and interventions scheduling for telecommunications". *Annals of Operations Research* 183, pp. 143–161.

Haughton, M. A. (2009). "An alternative tactic to deal with the contingency of driver absenteeism". *Journal of the Operational Research Society* 60.9, pp. 1207–1220.

Holland, C., J. Levis, R. Nuggehalli, B. Santilli, and J. Winters (2017). "UPS optimizes delivery routes". *Interfaces* 47.1, pp. 8–23.

Hollis, B. and P. Green (2012). "Real-life vehicle routing with time windows for visual attractiveness and operational robustness". *Asia-Pacific Journal of Operational Research* 29.04, p. 1250017.

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

Kovacs, A. A., S. N. Parragh, K. F. Doerner, and R. F. Hartl (2012). "Adaptive large neighborhood search for service technician routing and scheduling problems". *Journal of Scheduling* 15, pp. 579–600.

Lau, H. C. and A. Gunawan (2012). "The Patrol Scheduling Problem". *Practice and Theory of Automated Timetabling*.

Le Colleter, T., D. Dumez, F. Lehuédé, and O. Péton (2023). "Small and large neighborhood search for the park-and-loop routing problem with parking selection". *European Journal of Operational Research* 308.3, pp. 1233–1248.

Li, H., H. Wang, J. Chen, and M. Bai (2020). "Two-echelon vehicle routing problem with time windows and mobile satellites". *Transportation Research Part B: Methodological* 138, pp. 179–201.

Li, Y., A. Lim, and B. Rodrigues (2005). "Manpower allocation with time windows and job-teaming constraints". *Naval Research Logistics (NRL)* 52.4, pp. 302–311.

Lim, A., B. Rodrigues, and L. Song (2004). "Manpower allocation with time windows". *Journal of the Operational Research Society* 55.11, pp. 1178–1186.

Lin, S., V. Yu, and S. Chou (2009). "Solving the truck and trailer routing problem based on a simulated annealing heuristic". *Computers and Operations Research* 36, pp. 1683–1692.

Liu, M., D. Yang, Q. Su, and L. Xu (Sept. 2018). "Bi-objective approaches for home healthcare medical team planning and scheduling problem". *Computational and Applied Mathematics* 37 (4), pp. 4443–4474.

Lozano, L., D. Duque, and A. L. Medaglia (2016). "An exact algorithm for the elementary shortest path problem with resource constraints". *Transportation Science* 50.1, pp. 348–357.

Lozano, L. and A. L. Medaglia (2013). "On an exact method for the constrained shortest path problem". *Computers & Operations Research* 40.1, pp. 378–384.

Lozano, L. and J. C. Smith (2017). "A backward sampling framework for interdiction problems with fortification". *INFORMS Journal on Computing* 29.1, pp. 123–139.

Lübbecke, M. E. and J. Desrosiers (2005). "Selected topics in column generation". *Operations Research* 53.6, pp. 1007–1023.

Lund, K., O. B. Madsen, and J. M. Rygaard (1996). *Vehicle routing problems with varying degrees of dynamism*. IMM, Institute of Mathematical Modelling, Technical University of Denmark.

Ma, Z., C. Shao, Y. Song, and J. Chen (2014). "Driver response to information provided by variable message signs in Beijing". *Transportation research part F: traffic psychology and behaviour* 26, pp. 199–209.

Maghfiroh, M. F. and S. Hanaoka (2018). "Dynamic truck and trailer routing problem for last mile distribution in disaster response". *Journal of Humanitarian Logistics and Supply Chain Management* 8.2, pp. 252–278.

Marques, G., R. Sadykov, J.-C. Deschamps, and R. Dupas (2020). "An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem". *Computers & Operations Research* 114, p. 104833.

Martinez-Sykora, A., F. McLeod, C. Lamas-Fernandez, T. Bektaş, T. Cherrett, and J. Allen (2020). "Optimised solutions to the last-mile delivery problem in London using a combination of walking and driving". *Annals of Operations Research* 295.2, pp. 645–693.

Mathlouthi, I., M. Gendreau, and J.-Y. Potvin (2018). "Mixed integer linear programming for a multi-attribute technician routing and scheduling problem". *INFOR: Information Systems and Operational Research* 56.1, pp. 33–49.

Mathlouthi, I., M. Gendreau, and J.-Y. Potvin (2021a). "A metaheuristic based on Tabu search for solving a technician routing and scheduling problem". *Computers & Operations Research* 125, p. 105079.

Mathlouthi, I., M. Gendreau, and J.-Y. Potvin (2021b). "Branch-and-price for a multi-attribute technician routing and scheduling problem". *Operations Research Forum* 2, pp. 1–35.

Mendoza, J. E., A. L. Medaglia, and N. Velasco (2009). "An evolutionary-based decision support system for vehicle routing: The case of a public utility". *Decision Support Systems* 46.3, pp. 730–742.

Mendoza, J. E. and J. G. Villegas (2013). "A multi-space sampling heuristic for the vehicle routing problem with stochastic demands". *Optimization Letters* 7, pp. 1503–1516.

Misir, M., P. Smet, K. Verbeeck, and G. Vanden Berghe (2011). "Security personnel routing and rostering: a hyper-heuristic approach". *Proceedings of the 3rd International Conference on Applied Operational Research*. Vol. 3. Tadbir; Canada, pp. 193–205.

Mitrović-Minić, S. and G. Laporte (2004). "Waiting strategies for the dynamic pickup and delivery problem with time windows". *Transportation Research Part B: Methodological* 38.7, pp. 635–655.

Montoya, A., C. Guéret, J. E. Mendoza, and J. G. Villegas (2016). "A multi-space sampling heuristic for the green vehicle routing problem". *Transportation Research Part C: Emerging Technologies* 70, pp. 113–128.

Morris, E. A. and J. A. Hirsch (2016). "Does rush hour see a rush of emotions? Driver mood in conditions likely to exhibit congestion". *Travel Behaviour and Society* 5, pp. 5–13.

Mosquera, F., P. Smet, and G. V. Berghe (2019). "Flexible home care scheduling". *Omega* 83, pp. 80–95.

Naderi, B., M. A. Begen, G. S. Zaric, and V. Roshanaei (2023). "A novel and efficient exact technique for integrated staffing, assignment, routing, and scheduling of home care services under uncertainty". *Omega* 116, p. 102805.

Nowak, M. and P. Szufel (2024). "Technician routing and scheduling for the sharing economy". *European Journal of Operational Research* 314.1, pp. 15–31.

Paraskevopoulos, D. C., G. Laporte, P. P. Repoussis, and C. D. Tarantilis (2017). "Resource constrained routing and scheduling: Review and research prospects". *European Journal of Operational Research* 263.3, pp. 737–754.

Parragh, S. N. and J.-F. Cordeau (2017). "Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows". *Computers and Operations Research* 83, pp. 28–44.

Pekel, E. (2020). "Solving technician routing and scheduling problem using improved particle swarm optimization". *Soft Computing* 24.24, pp. 19007–19015.

Pereira, D. L., J. C. Alves, and M. C. de Oliveira Moreira (2020). "A multiperiod workforce scheduling and routing problem with dependent tasks". *Computers & Operations Research* 118, p. 104930.

Pessoa, A., R. Sadykov, and E. Uchoa (2018). "Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems". *European Journal of Operational Research* 270.2, pp. 530–543.

Pessoa, A., R. Sadykov, E. Uchoa, and F. Vanderbeck (2013). "In-Out separation and column generation stabilization by dual price smoothing". *Experimental Algorithms*. Ed. by V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 354–365. ISBN: 978-3-642-38527-8.

Pillac, V., C. Gueret, and A. L. Medaglia (2013). "A parallel matheuristic for the technician routing and scheduling problem". *Optimization Letters* 7.7, pp. 1525–1535.

Pillac, V., C. Guéret, and A. L. Medaglia (2012). "On the dynamic technician routing and scheduling problem". PhD thesis. Ecole des Mines de Nantes.

Pillac, V., C. Guéret, and A.-L. Medaglia (2018). "A fast reoptimization approach for the dynamic technician routing and scheduling problem". *Recent Developments in Metaheuristics*, pp. 347–367.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons.

Qiu, H., J. Wang, D. Wang, and Y. Yin (2023). "Service-oriented multi-skilled technician routing and scheduling problem for medical equipment maintenance with sudden breakdown". *Advanced Engineering Informatics* 57, p. 102090.

Rasmussen, M. S., T. Justesen, A. Dohn, and J. Larsen (2012). "The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies". *European Journal of Operational Research* 219.3, pp. 598–610.

Rastegar, M., H. Karimi, and H. Vahdani (2024). "Technicians scheduling and routing problem for elevators preventive maintenance". *Expert Systems with Applications* 235, p. 121133.

Reed, S., A. M. Campbell, and B. W. Thomas (2024). "Does parking matter? The impact of parking time on last-mile delivery optimization". *Transportation Research Part E: Logistics and Transportation Review* 181, p. 103391.

Restrepo, M. I., L. Lozano, and A. L. Medaglia (2012). "Constrained network-based column generation for the multi-activity shift scheduling problem". *International Journal of Production Economics* 140.1, pp. 466–472.

Rossit, D. G., D. Vigo, F. Tohmé, and M. Frutos (2019). "Visual attractiveness in routing problems: A review". *Computers & Operations Research* 103, pp. 13–34.

Rothenbächer, A.-K., M. Drexl, and S. Irnich (2018). "Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows". *Transportation Science* 52.5, pp. 1174–1190.

Russell, D., R. J. Rosati, P. Rosenfeld, and J. M. Marren (2011). "Continuity in home health care: is consistency in nursing personnel associated with better patient outcomes?" *Journal for Healthcare Quality* 33.6, pp. 33–39.

Sahoo, S., S. Kim, B.-I. Kim, B. Kraas, and A. Popov Jr (2005). "Routing optimization for waste management". *Interfaces* 35.1, pp. 24–36.

Sarasola, B., K. F. Doerner, V. Schmid, and E. Alba (2016). "Variable neighborhood search for the stochastic and dynamic vehicle routing problem". *Annals of Operations Research* 236, pp. 425–461.

Schrotenboer, A. H., E. Ursavas, and I. F. Vis (2019). "A branch-and-price-and-cut algorithm for resource-constrained pickup and delivery problems". *Transportation Science* 53.4, pp. 1001–1022.

Secomandi, N. (2001). "A rollout policy for the vehicle routing problem with stochastic demands". *Operations Research* 49.5, pp. 796–802.

Semet, F. (1995). "A two-phase algorithm for the partial accessibility constrained vehicle routing problem". *Annals of Operations Research* 61, pp. 45–65.

Shao, J., L. Kulik, E. Tanin, and L. Guo (2014). "Travel distance versus navigation complexity: A study on different spatial queries on road networks". *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1791–1794.

Shao, Y., J. F. Bard, and A. I. Jarrah (2012). "The therapist routing and scheduling problem". *IIE Transactions* 44.10, pp. 868–893.

Sheuerer, S. (2006). "A tabu search heuristic for the truck and trailer routing problem". *Computers and Operations Research* 33, pp. 894–909.

Soeffker, N., M. W. Ulmer, and D. C. Mattfeld (2024). "Balancing resources for dynamic vehicle routing with stochastic customer requests". *OR Spectrum*, pp. 1–43.

Solomon, M. M. (1987). "Algorithms for the vehicle routing and scheduling problems with time window constraints". *Operations Research* 35.2, pp. 254–265.

Srivatsa Srinivas, S. and M. Gajanand (2017). "Vehicle routing problem and driver behaviour: a review and framework for analysis". *Transport Reviews* 37.5, pp. 590–611.

Su, X., G. Xu, N. Huang, and H. Qin (2023). "A branch-and-price-and-cut for the manpower allocation and vehicle routing problem with staff qualifications and time windows". *Advanced Engineering Informatics* 57, p. 102093.

Tamke, F. and U. Buscher (2021). "A branch-and-cut algorithm for the vehicle routing problem with drones". *Transportation Research Part B: Methodological* 144, pp. 174–203.

Tricoire, F., N. Bostel, P. Dejax, and P. Guez (2013). "Exact and hybrid methods for the multiperiod field service routing problem". *Central European Journal of Operations Research* 21, pp. 359–377.

Tsang, E. and C. Voudouris (1997). "Fast local search and guided local search and their application to British Telecom's workforce scheduling problem". *Operations Research Letters* 20.3, pp. 119–127.

Uchoa, E., A. Pessoa, and L. Moreno (2024). *Optimizing with Column Generation: Advanced Branch-Cut-and-Price Algorithms (Part I)*. Tech. rep. L-2024-3. Cadernos do LOGIS-UFF, Universidade Federal Fluminense, Engenharia de Produção.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and M. Hennig (2019). "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests". *Transportation Science* 53.1, pp. 185–202.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and B. W. Thomas (2020). "On modeling stochastic dynamic vehicle routing problems". *EURO Journal on Transportation and Logistics* 9.2, p. 100008.

Ulmer, M. W., D. C. Mattfeld, and F. Köster (2018). "Budgeting time for dynamic vehicle routing with stochastic customer requests". *Transportation Science* 52.1, pp. 20–37.

Ulmer, M. W. and S. Streng (2019). "Same-day delivery with pickup stations and autonomous vehicles". *Computers & Operations Research* 108, pp. 1–19.

Ulmer, M. W. and B. W. Thomas (2018). "Same-day delivery with heterogeneous fleets of drones and vehicles". *Networks* 72.4, pp. 475–505.

Ulmer, M. W. and B. W. Thomas (2020). "Meso-parametric value function approximation for dynamic customer acceptances in delivery routing". *European Journal of Operational Research* 285.1, pp. 183–195.

Ulmer, M. W. (2017). *Approximate dynamic programming for dynamic vehicle routing*. Vol. 61. Springer.

Vidal, T., G. Laporte, and P. Matl (2020). "A concise guide to existing and emerging vehicle routing problem variants". *European Journal of Operational Research* 286.2, pp. 401–416.

Villegas, J., C. Prins, A. Medaglia, and N. Velasco (2010). "GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots". *Engineering Applications of Artificial Intelligence* 23, pp. 780–794.

Villegas, J., C. Prins, A. Medaglia, and N. Velasco (2013). "A matheuristic for the truck and trailer routing problem". *European Journal of Operational Research* 230, pp. 231–244.

Villegas, J., C. Prins, C. Prodhon, A. Medaglia, and N. Velasco (2011). "A GRASP with evolutionary path relinking for the truck and trailer routing problem". *Computers and Operations Research* 38, pp. 1319–1334.

Villegas, J., C. Guéret, J. E. Mendoza, and A. Montoya (June 2018). "The technician routing and scheduling problem with conventional and electric vehicle". working paper or preprint. URL: `https://hal.archives-ouvertes.fr/hal-01813887`.

Voccia, S. A., A. M. Campbell, and B. W. Thomas (2019). "The same-day delivery problem for online purchases". *Transportation Science* 53.1, pp. 167–184.

Xie, F., C. N. Potts, and T. Bektaş (2017). "Iterated local search for workforce scheduling and routing problems". *Journal of Heuristics* 23, pp. 471–500.

Yamín, D., A. L. Medaglia, and A. A. Prakash (2022). "Exact bidirectional algorithm for the least expected travel-time path problem on stochastic and time-dependent networks". *Computers & Operations Research* 141, p. 105671.

Yin, Y., D. Li, D. Wang, J. Ignatius, T. Cheng, and S. Wang (2023). "A branch-and-price-and-cut algorithm for the truck-based drone delivery routing problem with time windows". *European Journal of Operational Research* 309.3, pp. 1125–1144.

Yuan, B., R. Liu, and Z. Jiang (2015). "A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements". *International Journal of Production Research* 53.24, pp. 7450–7464.

Zamorano, E., A. Becker, and R. Stolletz (2018). "Task assignment with start time-dependent processing times for personnel at check-in counters". *Journal of Scheduling* 21, pp. 93–109.

Zamorano, E. and R. Stolletz (2017). "Branch-and-price approaches for the multi-period technician routing and scheduling problem". *European Journal of Operational Research* 257.1, pp. 55–68.

Zhan, Y. and G. Wan (2018). "Vehicle routing and appointment scheduling with team assignment for home services". *Computers & Operations Research* 100, pp. 1–11.

Zhang, J., K. Luo, A. M. Florio, and T. Van Woensel (2023). "Solving large-scale dynamic vehicle routing problems with stochastic requests". *European Journal of Operational Research* 306.2, pp. 596–614.

Zhang, J. and T. Van Woensel (2023). "Dynamic vehicle routing with random requests: A literature review". *International Journal of Production Economics* 256, p. 108751.

Zhou, H., H. Qin, C. Cheng, and L.-M. Rousseau (2023). "An exact algorithm for the two-echelon vehicle routing problem with drones". *Transportation Research Part B: Methodological* 168, pp. 124–150.

Zhou, L., S. Zhong, S. Ma, and N. Jia (2014). "Prospect theory based estimation of drivers' risk attitudes in route choice behaviors". *Accident Analysis & Prevention* 73, pp. 1–11.

Zhou, Y., M. Huang, H. Wu, G. Chen, and Z. Wang (2020). "Iterated local search with hybrid neighborhood search for workforce scheduling and routing problem". *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, pp. 478–485.

Appendix A. Labeling algorithm: In-depth description

In this section, we present a labeling algorithm to solve the pricing problem described in Section 2.3.2. Our algorithm is partially based on the ideas presented by Feillet et al. (2004) and Baldacci et al. (2011).

The algorithm extends labels ℓ (i.e., partial paths) starting at $\underline{0}$ and ending at a given node $n(\ell)$. A label ℓ stores the cumulative reduced cost $r(\ell)$, the cumulative time $t(\ell)$, the cumulative walking distance $w(\ell)$, the parking spot $p(\ell)$, a resource $\Pi_i^u \in \{0,1\}$ that takes the value of one if customer $i \in C$ cannot be visited by the partial path (due to time limit and walking distance constraints), and a resource $\Pi_i^{ng} \in \{0,1\}$ that takes the value of one if a customer $i \in C$ is forbidden by the ng-paths cycling restrictions described in Section 2.5.5. If the vehicle is not currently parked, $p(\ell) = -1$. Whenever a label has a resource consumption that exceeds the corresponding resource availability it is immediately discarded. Labels are also discarded if they do not comply with the ng-paths cycling restrictions or if they are dominated by other labels. We also discard labels using the rollback and the bounds pruning strategies described in Section 2.3.3. All the labels successfully extended to node $\overline{0}$ can be added to the MP as they represent feasible routes with negative reduced costs.

The main logic of the algorithm is presented in Algorithm 12. Line 1 runs the bounding procedure given the bound step size and the bounding time limits. To the

best of our knowledge, this is the first time that the pulse algorithm is included as a component of a labeling algorithm. Line 2 initializes the labels queue $\mathcal{Q}$ with a label ℓ representing the source depot (i.e., $n(\ell) = \underline{0}$) such that $r(\ell) = 0$, $t(\ell) = 0$, $w(\ell) = 0$, $p(\ell) = -1$, and $\Pi_i^u \in \{0,1\} = \Pi_i^{ng} \in \{0,1\} = 0$ for all customers $i \in \mathcal{C}$. Line 3 initializes the set of negative reduced cost routes $\mathcal{H}$. From lines 4 to 9, the algorithm retrieves and expands non-dominated labels until Υ routes with negative reduced cost have been found or until the queue of labels is empty. Line 5 retrieves the first label from the queue. Line 6 checks if the label is not dominated by another label at the same node. Line 7 checks if the label can be discarded using a bound on the best reduced cost that can be achieved given the current time consumption. Line 8 checks if the label can be discarded using the rollback pruning strategy. If the label is not discarded, line 9 extends the label through all the outgoing arcs of the current node and adds the new labels to $\mathcal{Q}$. Line 14 returns all the routes at the end depot with negative reduced cost stored in $\mathcal{H}$.

Algorithm 12 Labeling algorithm

Require: $\bar{\mathcal{G}}$, directed multi graph; ϕ, duration limit; ζ, walking distance limit; $\underline{0}$, start node; $\overline{0}$, end node; Δ, bound step size; $[\bar{t}, \underline{t}]$, bounding time limits; ℓ, initial label.

```
 1: bound(Ḡ, Δ, [t̄, t̲])                                   ▷ see §2.3.3
 2: push(Q, ℓ)
 3: H ← ∅
 4: while Q ≠ ∅ ∧ |H| ≤ Υ do
 5:     ℓ ←pop(Q)
 6:     if ¬dominance(ℓ) then                              ▷ see §4.7
 7:         if ¬bounds(n(ℓ), r(ℓ), t(ℓ)) then              ▷ see §2.3.3
 8:             if ¬rollback(n(ℓ), r(ℓ), t(ℓ), w(ℓ)) then  ▷ see §2.3.3
 9:                 Q ← Q ∪ extend(ℓ)                      ▷ see §4.7
10:             end if
11:         end if
12:     end if
13: end while
14: return H
```

A.1. Dominance rules

During the execution of the algorithm, several labels representing different partial paths will be expanded to a given node $i \in \mathcal{V}'$. With this in mind, we define dominance relations between two labels ℓ and ℓ' at a current node $n(\ell) = n(\ell')$. More precisely, label ℓ dominates label ℓ' if the following conditions hold:

$$p(\ell) = p(\ell'), \tag{30}$$

$$r(\ell) \leq r(\ell'), \tag{31}$$

$$t(\ell) \leq t(\ell'), \tag{32}$$

$$w(\ell) \leq w(\ell'), \tag{33}$$

$$\max\left\{\Pi_i^{ng}(\ell), \Pi_i^u(\ell)\right\} \leq \max\left\{\Pi_i^{ng}(\ell'), \Pi_i^u(\ell')\right\} \quad \forall i \in C. \tag{34}$$

Conditions (30) ensure that the two labels represent partial paths with equal parking status. Conditions (31)-(34) imply that any feasible extension of the dominated label ℓ' is also a feasible extension of label ℓ with an equal or better reduced cost. During the first iterations of the BPC procedure, is possible to ignore conditions (33)-(34) to accelerate the solution of the pricing problem.

The previous dominance rules are only valid if the master problem does not include a subset row inequality (SRI). To account for the presence of SRIs, we modified the dominance rules following the ideas presented by Jepsen et al. (2008). More specifically, let $\upsilon_S(\ell)$ denote the number of times modulo 2 that label ℓ has served the customers in $S \subseteq C$. When extending label ℓ through arc $(i, j) \in \bar{A}$ to create label ℓ', we set $\upsilon_S(\ell') = \upsilon_S(\ell)$ if $j \notin S$ and $\upsilon_S(\ell') = (\upsilon_S(\ell) + 1) \bmod 2$ otherwise. If, $\upsilon_S(\ell') = 0$ and $\upsilon_S(\ell) = 1$ we subtract the dual variable β_S from $r(\ell')$. Then, the dominance rules are modified by replacing condition 31 with:

$$r(\ell) - \sum_{\mathcal{S} \subseteq C | \upsilon_{\mathcal{S}}(\ell) > \upsilon_{\mathcal{S}}(\ell')} \beta_{\mathcal{S}} \leq r(\ell'), \tag{35}$$

Condition 35 ensures that $r(\ell)$ is sufficiently less than $r(\ell')$ to compensate for any possible additional penalties.

A.2. Label extension

A label is only extended if $t(\ell) \leq \phi$ and $w(\ell) \leq \zeta$. When a label ℓ is extended through arc $(i, j) \in \bar{\mathcal{A}}$ to create label ℓ' the resources are updated depending on the operation performed. There are four types of label extensions: (1) driving, (2) parking the vehicle and walking, (3) walking without unpark, and (4) walking with unpark. In what follows, we will describe the extension rules in detail for each case.

Case 1: *Driving*. In this case, the route is extended by driving between nodes i and j. This extension is only possible if $p(\ell) = -1$ and $(i, j) \in \mathcal{A}^1 \cup \mathcal{A}^3$ (i.e., through a driving arc). Label ℓ' has the following attributes:

$$p(\ell') = p(\ell), \tag{36}$$

$$r(\ell') = r(\ell) + r_{ij}, \tag{37}$$

$$t(\ell') = t(\ell) + \eta_{ij} + s_j, \tag{38}$$

$$w(\ell') = w(\ell), \tag{39}$$

$$\Pi^u_m(\ell') = \begin{cases} 0, & t(\ell') + \eta_{jm} + s_m \leq \phi \mid (j, m) \in \mathcal{A}^1; \\ 1, & \text{otherwise;} \end{cases} \qquad \forall m \in C, \tag{40}$$

$$\Pi_m^{ng}(\ell') = \begin{cases} 1, & (m = j) \vee (\Pi_m^{ng}(\ell) = 1 \wedge m \in \mathcal{N}_j); \\ 0, & \text{otherwise;} \end{cases} \qquad \forall m \in C. \tag{41}$$

Case 2: *Parking the vehicle and walking.* In this case, the route is extended by walking between nodes i and j after parking the vehicle at node i. This extension is only possible if $p(\ell) = -1$ and $(i, j) \in \mathcal{A}^2 \cup \mathcal{A}^4$ (i.e., through a walking arc). Label ℓ' has the following attributes:

$$p(\ell') = i, \tag{42}$$

$$r(\ell') = r(\ell) + r_{ij}, \tag{43}$$

$$t(\ell') = t(\ell) + \eta_{ij} + s_j, \tag{44}$$

$$w(\ell') = w(\ell) + \delta_{ij}, \tag{45}$$

$$\Pi_m^{u}(\ell') = \begin{cases} 0, & t(\ell') + \eta_{jm} + s_m + \eta_{mi} \leq \phi \mid (j, m), (m, i) \in \mathcal{A}^2; \\ 1, & \text{otherwise;} \end{cases} \qquad \forall m \in C, \tag{46}$$

$$\Pi_m^{ng}(\ell') = \begin{cases} 1, & (m = j) \vee (\Pi_m^{ng}(\ell) = 1 \wedge m \in \mathcal{N}_j); \\ 0, & \text{otherwise;} \end{cases} \qquad \forall m \in C. \tag{47}$$

Case 3: *Walking without unpark.* In this case, the route is extended by walking between nodes i and j. The vehicle remains parked at node $p(\ell)$. This extension is only possible if $p(\ell) \neq -1$ and $(i, j) \in \mathcal{A}^2 \cup \mathcal{A}^4$ (i.e., through a walking arc). Label ℓ' has the following attributes:

$$p(\ell') = p(\ell), \tag{48}$$

$$r(\ell') = r(\ell) + r_{ij}, \tag{49}$$

$$t(\ell') = t(\ell) + \eta_{ij} + s_j, \tag{50}$$

$$w(\ell') = w(\ell) + \delta_{ij}, \tag{51}$$

$$\Pi_m^u(\ell') = \begin{cases} 0, & t(\ell') + \eta_{jm} + s_m + \eta_{mp(\ell)} \leq \phi \mid (j,m),(m,p(\ell)) \in \mathcal{A}^2; \\ 1, & \text{otherwise;} \end{cases} \qquad \forall m \in C, \tag{52}$$

$$\Pi_m^{ng}(\ell') = \begin{cases} 1, & (m = j) \vee (\Pi_m^{ng}(\ell) = 1 \wedge m \in \mathcal{N}_j); \\ 0, & \text{otherwise;} \end{cases} \qquad \forall m \in C. \tag{53}$$

Case 4: *Walking with unpark*. In this case, the route is extended by walking between nodes i and j. The vehicle is recovered at node $p(\ell)$. This extension is only possible if the $p(\ell) \neq -1$ and $(i,j) \in \mathcal{A}^2 \cup \mathcal{A}^4$ (i.e., through a walking arc). Label ℓ_j has the following attributes:

$$p(\ell') = -1, \tag{54}$$

$$r(\ell') = r(\ell) + r_{ij} + \pi_j, \tag{55}$$

$$t(\ell') = t(\ell) + \eta_{ij}, \tag{56}$$

$$w(\ell') = w(\ell) + \delta_{ij}, \tag{57}$$

$$\Pi_m^u(\ell') = \begin{cases} 0, & t(\ell') + \eta_{jm} + s_m \leq \phi \mid (j,m) \in \mathcal{A}^1; \\ 1, & \text{otherwise;} \end{cases} \qquad \forall m \in C, \tag{58}$$

$$\Pi_m^{ng}(\ell') = \begin{cases} 1, & (m = j) \vee (\Pi_m^{ng}(\ell) = 1 \wedge m \in \mathcal{N}_j); \\ 0, & \text{otherwise;} \end{cases} \qquad \forall m \in C. \tag{59}$$

Appendix B. Detailed results for each PLRP instance

Table B.1 shows the performance of each algorithm on each instance of the VRPTR without carpooling. Each row corresponds to an instance. Columns 2 to 5 show the objective function of the solution found by the VNS, MSH, SLNS, and the BPC algorithm respectively. Column 6 shows the objective function of the current best-known solution.

Table B.2 reports the solutions found by MSH and BPC to the set of large instances. Each row corresponds to an instance. Column 2 reports the number of available workers k. Columns 3 and 4 show the objective function of the solution found by MSH and BPC. Column 5 reports the lower bound found by BPC.

Table B.1: Detailed results on the Coindreau et al. (2019) instances.

Instance	VNS	MSH	SLNS	BPC	BKS
20_A_1	32.4037	**30.9482**	**30.9482**	**30.9482**	30.9482
20_A_2	41.6349	**41.5547**	**41.5547**	**41.5547**	41.5547
20_A_3	39.4469	**36.2344**	**36.2344**	**36.2344**	36.2344
20_A_4	39.1459	**36.0251**	**36.0251**	**36.0251**	36.0251
20_A_5	**35.2747**	**35.2747**	**35.2747**	**35.2747**	35.2747
20_A_6	43.7627	**42.2866**	**42.2866**	**42.2866**	42.2866
20_A_7	40.1143	**38.6881**	**38.6881**	**38.6881**	38.6881
20_A_8	39.1395	**36.8504**	**36.8504**	**36.8504**	36.8504
20_A_9	29.9427	**29.6105**	**29.6105**	**29.6105**	29.6105
20_A_10	41.2404	**39.6656**	**39.6656**	**39.6656**	39.6656
30_A_1	41.2853	**40.7372**	**40.7372**	**40.7372**	40.7372
30_A_2	46.6485	45.6635	**45.6465**	**45.6465**	45.6465
30_A_3	50.9785	**49.1828**	**49.1828**	**49.1828**	49.1828
30_A_4	46.3310	**43.7556**	**43.7556**	**43.7556**	43.7556
30_A_5	48.3961	**47.0619**	**47.0619**	**47.0619**	47.0619
30_A_6	51.6505	**49.5880**	**49.5880**	**49.5880**	49.5880
30_A_7	45.5537	45.5537	45.5537	**45.5340**	45.5340
30_A_8	45.3691	**43.6799**	**43.6799**	**43.6799**	43.6799
30_A_9	41.8553	41.1629	41.1629	**41.1609**	41.1609
30_A_10	49.5241	**46.8936**	**46.8936**	**46.8936**	46.8936
40_A_1	**59.1695**	**59.1695**	**59.1695**	**59.1695**	59.1695
40_A_2	58.5186	58.5186	58.5186	**58.1191**	58.1191
40_A_3	63.1723	**62.9222**	**62.9222**	**62.9222**	62.9222
40_A_4	**50.3730**	**50.3730**	**50.3730**	**50.3730**	50.3730
40_A_5	52.7504	**51.7278**	**51.7278**	**51.7278**	51.7278
40_A_6	63.0658	**61.2621**	**61.2621**	**61.2621**	61.2621
40_A_7	57.7673	54.8824	55.6158	**54.8554**	54.8554
40_A_8	56.3737	56.3140	**55.9774**	**55.9774**	55.9774
40_A_9	56.4612	56.3871	56.3871	**56.3013**	56.3013
40_A_10	**57.6954**	**57.6954**	**57.6954**	**57.6954**	57.6954
50_A_1	60.5063	**57.0235**	**57.0235**	**57.0235**	57.0235
50_A_2	62.4825	62.1686	60.6031	**60.5906**	60.5906
50_A_3	65.8727	63.8855	63.6788	**63.5529**	63.5529
50_A_4	58.2333	57.9472	56.9391	**56.6342**	56.6342
50_A_5	64.1751	**64.0909**	**64.0909**	**64.0909**	64.0909
50_A_6	66.1254	65.0105	65.0105	**64.8116**	64.8116
50_A_7	63.6674	63.6674	63.6674	**63.6350**	63.6350
50_A_8	70.9057	69.7911	68.8367	**68.6323**	68.6323
50_A_9	**58.2212**	58.7191	**58.2212**	**58.2212**	58.2212
50_A_10	**60.7796**	61.2239	**60.7796**	**60.7796**	60.7796

Table B.2: Detailed results on the PLRP set of large instances.

Instance	k	MSH	BPC	LB
60_A_1	5	63.6180	63.6180	61.6495
60_A_2	5	67.9332	64.8337	63.1645
60_A_3	5	66.7018	64.3194	63.3934
60_A_4	5	69.4019	68.8561	68.8561
60_A_5	5	72.0505	67.8520	67.4220
60_A_6	5	65.7642	65.6751	64.3115
60_A_7	5	65.4693	64.7246	64.7246
60_A_8	5	66.4684	65.6574	65.6574
60_A_9	5	69.3708	67.1968	65.3779
60_A_10	6	77.4000	77.3963	74.5644
70_A_1	6	80.7800	80.3145	77.4835
70_A_2	5	85.1100	85.1052	81.5012
70_A_3	6	71.4200	71.4182	69.7106
70_A_4	6	74.8472	74.4721	71.8464
70_A_5	5	79.0669	79.0669	76.9217
70_A_6	6	70.8563	70.8563	69.3213
70_A_7	6	78.2697	78.2697	76.0983
70_A_8	6	75.8930	75.8930	71.8024
70_A_9	5	80.0280	79.8592	75.8490
70_A_10	6	82.5269	82.1809	80.0585
80_A_1	6	92.0970	91.4923	86.1695
80_A_2	6	92.2233	90.9719	86.9094
80_A_3	6	96.9931	96.2431	90.0295
80_A_4	6	99.6504	94.4960	89.9932
80_A_5	6	93.4427	93.4428	86.3854
80_A_6	6	94.6270	91.1502	87.4419
80_A_7	6	90.1759	90.1759	86.4039
80_A_8	6	92.4352	92.4352	86.7532
80_A_9	6	101.2854	101.0565	92.7555
80_A_10	6	93.1675	92.9522	88.3396
90_A_1	7	92.1341	92.1341	88.6969
90_A_2	7	85.6971	85.6971	80.8916
90_A_3	7	88.6107	88.6107	81.9134
90_A_4	7	94.0196	91.9519	87.9134
90_A_5	7	100.7159	100.3967	94.3991
90_A_6	7	81.6312	81.5321	76.5057
90_A_7	7	95.2463	95.2463	88.8707
90_A_8	7	97.8633	96.7539	91.6895
90_A_9	7	85.2391	85.2391	78.1943
90_A_10	7	99.6994	98.5392	92.9275

Appendix C. New best known solutions for the no-team STRSP

Table C.1 compares the solution found by BPC and the best solution reported in the literature. Column 1 shows the instance identifier. Column 2 shows the objective function of the best known solution reported by Gu et al. (2022). Column 3 reports the objective function of the solution found by BPC.

Table C.1: New best known solutions for the no-team STRSP 100-task instances.

Instance	L-ILS	BPC
RC101_C_5x4_100	1658.37	1654.80
RC101_C_6x6_100	1654.98	1644.32
R103_C_7x4_100	1335.65	1325.31
C101_R_5x4_100	5587.52	5572.99
RC101_R_5x4_100	4829.37	4764.44
C103_R_6x6_100	4804.61	4799.01
R101_R_6x6_100	5945.14	5944.91
RC101_R_6x6_100	4903.70	4835.20
C101_R_7x4_100	5241.93	5208.30
C103_R_7x4_100	1980.72	1940.21
R103_R_7x4_100	2104.89	2104.17
RC103_R_7x4_100	2585.95	2568.25

Appendix D. Relative effectiveness of the pruning strategies

We designed an experiment to assess the relative contribution of each pruning strategy to the overall performance of the pulse algorithm (PA) used to solve the pricing problem. This experiment consists of solving every instance of the WSRP-PL described in Section 3.5.1 using two versions of the PA. In the first version, the PA uses a one-time pre-processing procedure to compute lower bounds on the minimum reduced cost from any node to the end node. Then, condition (3.50) is used to prune partial paths. We label this version of the PA, as PA-OTB. On the contrary, in the second version, the PA computes the lower bounds on the fly after deciding which workers are assigned to the team. Then, condition (3.51) is used to prune partial paths. We denote this version as PA-OTF.

Figures D.1 and D.2 show the relative effectiveness of the pruning strategies as the number of customers varies when using PA-OTB and PA-OTF. Regardless of the version of the PA used, the most effective strategy is pruning by infeasibility. More specifically, the algorithm relies on this strategy to prune almost 67% and 57% of the paths, followed by pruning by bounds, and rollback. Note however, that there is an increase of up to 15% on the relative effectiveness of the bounds pruning strategy when condition (3.51) is used. This result can be explained by an increase in the quality of the lower bounds used by the PA-OTF.

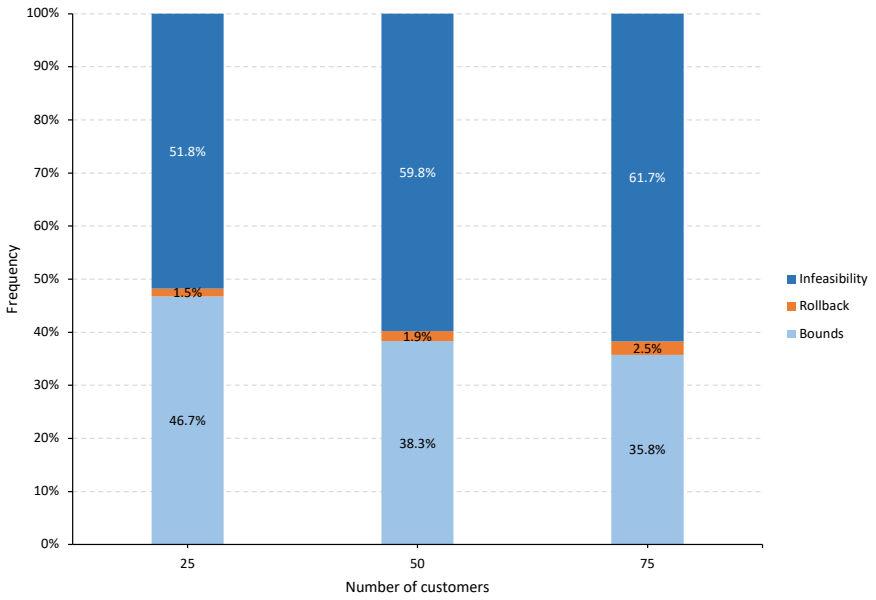


Figure D.1: Relative effectiveness of the pruning strategies, when using PA-OTB.

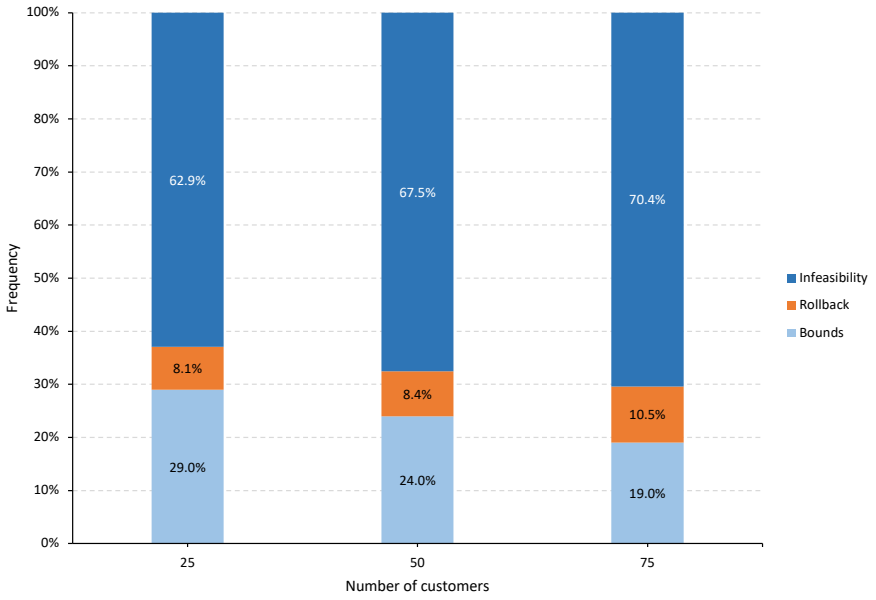


Figure D.2: Relative effectiveness of the pruning strategies, when using PA-OTF.