FEDERAL UNIVERSITY OF SÃO CARLOS
CENTRE OF EXACT SCIENCES AND TECHNOLOGY
GRADUATE PROGRAM IN PRODUCTION ENGINEERING

Alex Paranahyba de Abreu

The robust pickup and delivery problem with time windows

São Carlos - SP, Brazil
2025

Alex Paranahyba de Abreu

The robust pickup and delivery problem with time windows

Master's dissertation presented to the Graduate Program in Production Engineering of the Federal University of São Carlos as a partial requirement to obtain the degree of Master in Production Engineering.

Supervisor: Prof. Dr. Pedro A. Munari Jr.
Co-supervisor: Prof. Dr. Luciano C. A. da Costa

Funding: FAPESP and CAPES.

São Carlos - SP, Brazil
2025

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Alex Paranahyba de Abreu, realizada em 29/01/2025.

Comissão Julgadora:

Prof. Dr. Pedro Augusto Munari Junior (UFSCar)

Prof. Dr. Luciano Carlos Azevedo da Costa (UFPB)

Prof. Dr. Anand Subramanian (UFPB)

Prof. Dr. Maria Battarra (Bath)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Engenharia de Produção.

À minha família:
Walfredo, Valéria, Douglas, Aline e Débora.

Acknowledgements

First and foremost, I would like to express my deep gratitude to my family: my father, Walfredo, my mother, Valéria, my brother, Douglas, my sister-in-law, Aline, and my girlfriend, Débora. Their unconditional support, constant encouragement, and constant trust in me have been fundamental to the completion of this master's degree.

I extend my sincere gratitude to my supervisors, Prof. Dr. Pedro Munari and Prof. Dr. Luciano Costa, for their guidance, patience, and invaluable teachings throughout this journey. A special thank you to Prof. Dr. Maria Battarra for welcoming me during my research internship abroad and for so generously sharing her knowledge. I am truly grateful for all the opportunities you have provided me with and for all the learning that I will carry with me throughout my personal and professional journey.

I would also like to thank my friends and colleagues from the Graduate Programme in Production Engineering, especially those from the Operational Research Group, for their companionship, the exchange of knowledge, and the moments we have shared along this path.

Finally, I would like to thank the São Paulo Research Foundation (FAPESP), Brazil (grant numbers 2023/08678-8 and 2022/10993-6), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil (Finance Code 001) for funding this research.

Agradecimentos

Em primeiro lugar, gostaria de expressar minha profunda gratidão à minha família: meu pai, Walfredo, minha mãe, Valéria, meu irmão, Douglas, minha cunhada, Aline, e minha namorada, Débora. O apoio incondicional, o incentivo constante e a confiança que sempre depositaram em mim foram fundamentais para a realização deste mestrado.

Aos meus orientadores, Prof. Dr. Pedro Munari e Prof. Dr. Luciano Costa, minha sincera gratidão pela orientação, paciência e ensinamentos ao longo desta jornada. Um agradecimento especial à Profa. Dra. Maria Battarra, por me receber durante meu estágio de pesquisa no exterior e por compartilhar seus conhecimentos de forma tão generosa. Sou muito grato por todas as oportunidades que vocês me proporcionaram e por todo o aprendizado que levarei comigo ao longo de minha trajetória pessoal e profissional.

Agradeço também aos amigos e colegas do Programa de Pós-Graduação em Engenharia de Produção, especialmente aos do Grupo de Pesquisa Operacional, pelo companheirismo, pela troca de conhecimentos e pelos momentos compartilhados ao longo desta caminhada.

Agradeço à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (processos 2023/08678-8 and 2022/10993-6) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil (código de financiamento 001) pelo financiamento desta pesquisa.

Abstract

This study addresses the robust pickup and delivery problem with time windows (RPDPTW), in which uncertainty in demands and travel times is modelled using robust optimisation. The RPDPTW involves determining the least-cost routes to serve transportation requests from origins to destinations, while respecting vehicle capacity and time window constraints under all anticipated realisations of uncertain data. Two robust counterpart formulations are proposed. The first employs the linearisation of recursive equations to produce a compact formulation, suitable for implementation with general-purpose mixed-integer programming solvers. The second relies on a cutting-plane approach, incorporated into a tailored branch-and-cut (B&C) algorithm. Extensive computational experiments were conducted to evaluate the proposed methods across diverse instance configurations. While the compact formulation performed effectively in some cases, the B&C algorithm solved a larger number of instances to optimality and consistently provided high-quality solutions under uncertainty. Robust solutions significantly reduced the risk of infeasibility compared to deterministic solutions, with modest increases in routing costs. In addition, we provide managerial insights by analysing the results of Monte Carlo simulations across different instance characteristics.

Keywords: vehicle routing, pickup and delivery problem, time windows, demand uncertainty, travel time uncertainty.

Resumo

Este estudo aborda o problema de coletas e entregas com janelas de tempo robusto (RPDPTW, do inglês robust pickup and delivery problem with time windows), no qual a incerteza nas demandas e nos tempos de viagem é modelada por meio da otimização robusta. O RPDPTW consiste em determinar rotas de menor custo para atender às requisições de transporte, levando cargas de origens a destinos, respeitando restrições de capacidade de veículos e de janelas de tempo considerando todas realizações antecipadas dos dados incertos. Duas formulações robustas foram propostas. A primeira emprega a técnica de linearização de equações recursivas resultando em uma formulação compacta, adequada para ser implementada com resolvedores de programação inteira mista de propósito geral. A segunda baseia-se em uma abordagem de planos de corte, sendo incorporada a um algoritmo especializado do tipo branch-and-cut (B&C). Experimentos computacionais extensivos foram realizados para avaliar os métodos propostos em diversas configurações de instâncias. Embora a formulação compacta tenha se mostrado eficiente em alguns casos, o algoritmo B&C solucionou um maior número de instâncias até a otimalidade e forneceu, de forma consistente, soluções de alta qualidade sob incerteza. As soluções robustas obtidas com os métodos reduziram significativamente o risco de infactibilidade em comparação com soluções determinísticas, com aumentos modestos nos custos de transporte. Além disso, foram fornecidos insights gerenciais analisando os resultados das simulações de Monte Carlo em diferentes características de instância.

Palavras-chave: roteamento de veículos, problema de coletas e entregas, janelas de tempo, demanda incerta, tempo de viagem incerto.

List of Figures

List of Tables

List of Algorithms

Summary

Chapter 1
Introduction

The *pickup and delivery problem* (PDP) is a generalisation of the *vehicle routing problem* (VRP), which consists of determining the least-cost plan of routes for a fleet of vehicles to serve a set of customers while considering operations of collection and distribution of goods (Battarra; Cordeau; Iori, 2014). Variants of the PDP are commonly classified into one-to-many-to-one (1-M-1), many-to-many (M-M), and one-to-one (1-1) structures, based on the nature of customer demands (Berbeglia et al., 2007). These classifications address distinct logistics contexts, such as product transportation, inventory balancing, and services like meal delivery. Each variant may incorporate different features, including time windows, split loads, or transshipment operations, reflecting real-world operational requirements (Zang et al., 2022).

Our study focuses on the 1-1 PDP, where each transportation request is characterized by a single origin (pickup) and a single destination (delivery) location. In particular, we address the variant in which locations must be visited within a specified time interval for service, known as the *pickup and delivery problem with time windows* (PDPTW). The time windows are hard constraints, and vehicles are allowed to wait if they arrive at a location before the time window opens. The PDPTW aims to find least-cost routes that service the set of transportation requests exactly once while respecting the pairing and precedence relationships between pickup and delivery locations, as well as vehicle capacity and customer time windows (Dumas; Desrosiers; Soumis, 1991).

Although most publications on PDPTW variants assume that all parameters are known and deterministic, real-world data is typically uncertain and affected by estimation errors. Many methods have been proposed to address uncertainties in routing problems, particularly stochastic programming (Gendreau; Jabali; Rei, 2016) and *robust optimisation* (RO) (Ordóñez, 2010). Unlike stochastic programming approaches, RO methods do not require complete knowledge of the probabilistic distribution of the uncertain parameters. Instead, they rely on the estimated worst-case values of the parameters to provide feasible solutions, even in the presence of data uncertainty.

Ensuring feasibility for all realisations of an uncertain parameter may result in conservative solutions with high costs. Therefore, modelling an appropriate *uncertainty set* based on the decision maker's risk aversion helps to avoid an overly pessimistic evaluation. Different uncertainty sets have been applied in the context of *robust VRPs* (RVRPs) (Subramanyam; Repoussis; Gounaris, 2020; Wang; Subramanyam; Gounaris, 2022). In particular, a widely studied approach is the polyhedral *cardinality-constrained* uncertainty set, which limits the number of uncertain parameters allowed to simultaneously attain

their worst-case values (Bertsimas; Sim, 2004).

Tractable robust counterpart formulations can be derived using the so-called dualisation technique in the presence of polyhedral uncertainty sets. This approach is common among RVRP variants, although it often requires additional constraints, variables, or even deep reformulations (Agra et al., 2012; Gounaris; Wiesemann; Floudas, 2013; De La Vega; Munari; Morabito, 2019). This method has been frequently outperformed by the linearisation scheme, which involves incorporating recursive equations into compact vehicle flow formulations (Munari et al., 2019; Campos; Coelho; Munari, 2024).

1.1 Contributions

While the literature on the RVRP is extensive, RO methods for PDP variants have been scarcely explored. A different robust variant of the PDPTW has been addressed in the study by Liu et al. (2023), where the authors consider electric vehicles, demand uncertainty, and a scenario-based uncertainty set. To our knowledge, our study is the first to address the *robust pickup and delivery problem with time windows* (RPDPTW). Both demands and travel times are considered uncertain and are modelled using the cardinality-constrained uncertainty set. The main contributions of this study are as follows:

- We propose three robust counterpart formulations to ensure robustness for capacity and time windows. The first approach consists in compact formulations, i.e., models with polynomial numbers of variables and constraints. The second approach relies on cutting planes, resulting in a formulation with an exponential number of constraints.
- We design a tailored *branch-and-cut* (B&C) algorithm based on the cutting-plane formulation. This method ensures robustness for capacity and time windows by dynamically introducing constraints. We develop robust separation procedures to identify the worst-case load and service start time.
- We adapt deterministic benchmark instances to accommodate uncertainties. The best-known solutions for these instances are presented for the addressed variants.
- We provide computational evidence demonstrating that our robust approach achieves a high feasibility rate with minimal additional costs. We conduct a series of Monte Carlo simulations with varying input data to assess the risk of robust solutions becoming infeasible when implemented.

1.2 Study organisation

This study is organised as follows. Chapter 2 reviews the literature related to our study. In Chapter 3, we formally define the deterministic and robust problems. The proposed robust counterpart formulations are presented in Chapter 4 along with the details

of the B&C algorithm, including its configuration and the separation procedures. The experimental design is described in Chapter 5, along with the computational results and robustness analysis. Chapter 6 summarises our conclusions and outlines opportunities for future research. Additionally, the Appendix A presents a compact formulation based on the classical dualisation technique and its computational results.

Chapter 2
Literature Review

In this chapter, we discuss the existing literature on PDPs and RO for routing problems, the two main aspects of this study. In Section 2.1, we detail important PDP variants, show how they have been solved, and summarise recent surveys on the topic. Then, Section 2.2 shows the basis of RO methods and how they have been used to solve routing problems under uncertainty and, specially, PDP variants.

2.1 Pickup and delivery problems

The early effort to develop exact methods for the PDPTW was made by Dumas, Desrosiers and Soumis (1991). They propose a branch-and-price algorithm to solve the problem but also present a compact formulation which can be implemented in general-purpose MIP solvers. The research on routing problems with pickup and delivery operations is extensive and surveys have focused on different features of the PDP. Early reviews on VRP with pickups and deliveries are provided by Savelsbergh and Sol (1995) and Mitrovic-Minic (1998).

The review of Berbeglia et al. (2007) grouped the PDP variants into three main categories based on the structure of the demands. The one-to-many-to-one (1-M-1) problem is illustrated by the transportation of final products to customers and collecting supplies back to the factory. Many-to-many (M-M) problems in which each load has multiple origin and destination and could be illustrated by the operation of balancing the inventory of multiple factories. Finally, in one-to-one (1-1) problems, each load (request) has a specific origin and destination and its application is found, e.g., in meal delivery services. A variant of the 1-1 is the dial-a-ride problem (DARP) which addresses people transportation in applications including school bus routing and car pooling and is surveyed by Cordeau and Laporte (2007) and Doerner and Salazar-González (2014).

Another way of dividing the literature on PDP is onto two main classes. The survey of Parragh, Doerner and Hartl (2008a) details the first class (transportation from/to a depot), which is a 1-M-1 setting, and shows that it can be subdivided into four more classes. The first two classes are characterised by single demand customers - each customer presents either pickup or delivery demands - while customers are allowed to present both demands in the remainder two classes. The first class is the VRP with clustered backhauls (VRPCB) in which all delivery customers must be served first. In the VRP with mixed linehauls and backhauls (VRPMB), the second class, the vehicle is allowed to serve a mixed

sequence of delivery and pickup customers. The VRP with divisible delivery and pickup (VRPDDP) allows the vehicle to visit more than once each customer which has both type of demands. Finally, in the VRP with simultaneous delivery and pickup (VRPSDP), each customer demands must be satisfied in only one vehicle visit.

The second class (transportation between customers) is addressed by Parragh, Doerner and Hartl (2008b) which subdivides it into pickup and delivery VRP (PDVRP), which is the M-M setting, the pickup and delivery problem (PDP), which is the 1-1 setting, and the dial-a-ride problem (DARP). For both classes, a single vehicle situation changes the classification from VRP to TSP with the exception of PDP and DARP which becomes SPDP, for single-vehicle PDP, and SDARP, for single-vehicle DARP. Developments that consider the single vehicle routing problem with pickups and deliveries can be seen in Kalantari, Hill and Arora (1985), Hernández-Pérez and Salazar-González (2004), Hernández-Pérez and Salazar-González (2007), Gribkovskaia and Laporte (2008), Battarra et al. (2010), Erdoğan et al. (2012), among others.

In Savelsbergh and Sol (1995) there is a review of methods applied to different variants of the PDP and they derived a general formulation that addresses all PDP variants. Their model handles features such as each vehicle can present its own origin and destination locations and also each request can present more than one pickup location with different demands (respect. delivery locations and demands). Also, for an effort towards a generalised model able to address many problem variants, the paper of Praxedes et al. (2024) presents an exact method for several problems with simultaneous pickup and delivery demands.

Since the survey publications presented previously provide different notation and classification, it worth noticing that the differentiation between 1-M-1, M-M, and 1-1 is more common in the recent literature (Battarra; Cordeau; Iori, 2014). Therefore, this classification is used hereafter and the use of 'PDP' refers to all routing problem variants in which operations of collection and distribution is performed. The reader interested in surveys on specific PDP variants is referred to Koç and Laporte (2018), for 1-M-1 problems, Koç, Laporte and Tükenmez (2020), for the PDP with simultaneous pickup and delivery, Cordeau, Laporte and Ropke (2008), for 1-1 problems, Berbeglia, Cordeau and Laporte (2010) and Cai et al. (2023), for dynamic problems.

Exact methods have been proposed for the 1-1 PDP. The early study of Dumas, Desrosiers and Soumis (1991) proposed a branch-and-price algorithm. They also presented a compact formulation for the problem using the three-index VRP formulation. Another branch-and-price algorithm was developed in Savelsbergh and Sol (1998) considering an applied routing problem. They present the method for the generalised PDP and solve 1-1 PDP instances and also a dynamic variant. Lu and Dessouky (2004) addressed a PDP variant in which heterogeneous vehicles are based in different depots through a branch-and-cut algorithm that relies on a polynomial size two-index formulation with

additional variables to indicate precedence. Their method incorporate valid inequalities for this additional variable. Two branch-and-cut methods were developed by Ropke, Cordeau and Laporte (2007). The first algorithm relied on a formulation with an exponential number of pairing constraints and the second formulation presents pairing, capacity, and time windows constraints in an exponential size. They show that some valid inequalities for the DARP and VRPTW are also valid for the PDPTW and propose new ones.

The branch-and-cut-and-price algorithm of Ropke and Cordeau (2009) rely on two different shortest path problems for the pricing subproblem in which the first solves an mixed linear programming model and the second solves its relaxation. To solve these subproblems they apply a branch-and-cut method with innovative valid inequalities. Baldacci, Bartolini and Mingozzi (2011) proposes an exact method that extends the framework for the CVRP by proposing new bounding procedures, cut-and-column generation method, and dynamic programming algorithm for generating elementary routes. Furtado, Munari and Morabito (2017) proposed a compact two-index formulation that ensures the pairing and precedence of pickup and delivery nodes with a continuous variable. The proposed model outperforms the 2-index model of Lu and Dessouky (2004) and 3-index formulation as presented in Desaulniers et al. (2002). Gschwind et al. (2018) develop a bidirectional labelling column generation method which brings more efficiency to previously proposed branch-and-cut-and-price methods. However, they show that their results are still comparable with the one of Baldacci, Bartolini and Mingozzi (2011). The branch-and-cut-and-price solver proposed by Pessoa et al. (2020) for routing problems is also tested in PDPTW instances.

Heuristic and metaheuristic algorithms for this problem have also been proposed. In fact, one can see that these approaches are more common in the PDPTW literature in comparison to exact methods due to the difficulty of solving the problem. Nanry and Barnes (2000) pioneered this approach by developing a reactive tabu search algorithm. This algorithm was improved by Lau and Liang (2002) which proposed a hybrid insertion heuristic. Also considering the tabu search, Li and Lim (2001) combined it with a simulated annealing. A grouping genetic algorithm was proposed by Pankratz (2005). Lu and Dessouky (2006) introduced an insertion-based heuristic which also considers the increment in the travel time along with the reduction in the time window slack. A two-stage algorithm that first minimise the number of routes and then decrease the routing cost was developed by Bent and Van Hentenryck (2006) using a simulated annealing and a large neighbourhood search. Ropke and Pisinger (2006) applied multiple heuristics to remove customers and reinsert them in the solution. The guided ejection search was proposed by Nagata and Kobayashi (2010). Curtois et al. (2018) propose a metaheuristic that combines the local search, the large neighbourhood search, and the guided ejection search. A ruin and recreate algorithm was developed by Christiaens and Berghe (2020) for several CVRP variants, including the PDPTW. Recently, Sartori and Buriol (2020) extended previous researches incorporating them with the large neighbourhood search and a set partitioning PDPTW

formulation in an iterated local search manner.

Different features have been also incorporated into the 1-1 PDP such as different vehicle depot location (Lu; Dessouky, 2004), split loads (Nowak; Ergun; White, 2008), heterogeneous fleet (Qu; Bard, 2015), shuttle routes (Masson et al., 2014) ride times (Gschwind, 2015), demand handling considerations (multiple stacks) (Cherkesly et al., 2016), integration with scheduled lines (Ghilas et al., 2018), requests with one delivery node and more than one pickup node (Naccache; Côté; Coelho, 2018), electric vehicles (Goeke, 2019), occasional drivers (Dahle et al., 2019), first-in-last-out loading (Alyasiry; Forbes; Bulmer, 2019), detachable trailers (Drexl, 2021), multiple products (Gasque; Munari, 2022), transshipment operations (Lyu; Yu, 2023), and drone assistance (Meng; Chen; Li, 2024).

2.2 Robust optimisation

Robust optimisation was introduced by Soyster (1973) to formulate models that lead to solutions that are immune to data uncertainty without the necessity of complete knowledge of the probability distribution. Their method ensures robustness by incorporating into the deterministic formulation the worst-case value of uncertain parameters. This approach guarantees feasibility for all possible variation of parameters, however it leads to overly conservative solutions with poor objective function value.

The first RO approach introduced for the VRP was carried out by Sungur, Ordóñez and Dessouky (2008). Their method also resulted in overly conservative solutions since it was developed to guarantee feasibility for the overall worst-case scenario, in which all uncertain parameters attain their worst-case value.

To mitigate overly pessimistic solutions when considering all possible data realisations, recent RO approaches consider a support uncertainty set which models all important realisations of the uncertain parameter. This set can be modelled in several different ways such as an ellipsoidal set where the parameters values ranges within a ellipsoidal representation based on a demand covariance matrix (Ben-tal; Nemirovski, 1999), a knapsack set where the parameters are previously grouped and the deviation from the expected value is limited (Gounaris; Wiesemann; Floudas, 2013), a factor model set which represents the deviation as a linear combination of independent factors within a factor loading matrix (Ceria; Stubbs, 2006), a discrete set where a convex hull of the possible realisations is defined by the decision-maker (Solano-charris; Prins; Santos, 2015), and finally a cardinality-constrained set where up to a predefined number of parameters reach their worst-case (Bertsimas; Sim, 2004).

Gounaris, Wiesemann and Floudas (2013) derived seven robust counterpart formulations for the CVRP under demand uncertainty considering the knapsack and the factor model uncertainty sets in which they applied Bertsimas and Sim (2004)'s dualisation

technique to obtain the counterparts. They also show how to incorporate demand uncertainty on the Rounded Capacity Inequalities (RCI) to add as valid inequalities separated heuristically in a Tabu Search fashion.

To consider travel time uncertainty, Agra et al. (2012) relied on a 3-index formulation to model a layered graph and create a 4-index binary variable which shows if a vehicle k services a node i in a position l of its path just before servicing node j. Then, they use this variable to incorporate robustness for time window constraints considering the cardinality constrained uncertainty set and using also the dualisation technique to obtain the robust counterpart.

A variant of the VRP under both demand and travel time uncertainty was addressed by Lee, Lee and Park (2012) which proposed a decomposition approach with a dynamic programming algorithm for the subproblem to handle the uncertain data. They modelled both demand and travel time realisations with cardinality-constrained uncertainty sets. However, their problem considers that each customer is associated with a deadline instead of a time windows.

Dualisation approaches for demand and travel time uncertainties were outperformed by the linearisation of recursive equations introduced by Munari et al. (2019). Their method uses the fact that, given a route solution, one can check if the capacity or the time window constraints were violated via recursive equations. This approach has been broadly incorporated in a broad variety of robust optimisation problems such as routing, scheduling, and team orienteering (De La Vega; Munari; Morabito, 2020; Yu; Cheng; Zhu, 2022).

However, robust variants of the PDP have been barely addressed in the literature. To our knowledge, the PDPTW has never been addressed using RO as presented in this study, thus in what follows we present a review on related extensions of the PDP and PDPTW, mostly addressing rich variants. Tajik et al. (2014) introduces RO techniques for a rich 1-M-1 PDP variant. The addressed problem considers soft time windows and backhaul constraints. The paper depicts an environmental approach of routing problems considering in the objective function pollution costs as well as mechanical aspects. The authors consider that service times, travel times, fuel consumption and greenhouse gas emission costs are uncertain and described in a bounded box uncertainty set.

Further studies that consider 1-M-1 PDP variants via RO modelling have addressed the problem with several additional features. Golsefidi and Jokar (2020) consider the integrated decisions of production, inventory, distribution, and routing operations. To address a real-world approach, the retailer's demand (i.e., delivery demand), the amount of defective products (i.e., pickup demand), and the reproduction cost are considered as uncertain parameters and are modelled through a cardinality-constrained uncertainty set. Santos et al. (2020) solve the VRP with selective backhauls (VRPSB) using a branch-and-cut method. They considered that the revenue is a uncertain parameter and implemented

budget and factor model support sets. Vakili, Shirazi and Gitinavard (2021) address the Green Open Location-Routing Problem with Simultaneous Pickup and Delivery to optimise costs of fuel consumption, greenhouse gases emission, and warehouse location costs. They resort to a scenario-based setting considering both demands and travel time uncertain. Finally, Pavlova and Shichiyakh (2021) address the VRPTW with workforce allocation and simultaneous pickup and delivery. Travelling costs and times, demands, service time, and labor wage are considered uncertain and modelled using discrete scenarios.

Regarding the M-M PDP variants, Huang et al. (2020) modelled freight shipping services between regions separated by a major ocean based on the real-world hub-and-spoke maritime network of Bohai Sea, area of China. Demands and costs are considered uncertain and defined in a discrete scenario-based uncertainty set. Allahyari, Yaghoubi and Woensel (2021) consider the transportation of valuable goods in which security carriers aim to reduce robbery risk along with operational costs. The uncertain demands are modelled as a hard worst-case in which all uncertain parameters assume its worst-case leading to an overly conservative solution with a poor objective function. Hansuwa, Kumar and Chandrasekharan (2022) studied the vehicle routing problem with hard time windows and simultaneous pickup and delivery. They modelled uncertain travel and service times and demands in box and ellipsoidal uncertainty sets. For the box uncertainty set, they obtained linear formulations with the techniques of dualisation and linearisation of recursive equations.

Notice that the majority of publications on robust variants of the PDP mainly focused on 1-M-1 and M-M rich problems. Two studies have explored a robust 1-1 PDP variant combined with the 1-M-1 structure, where the depot can be paired with customers, acting as their corresponding pickup or delivery node. Chami et al. (2018) and Chami et al. (2022) address this variant, incorporating time windows, selective requests, and profit maximisation. They consider uncertain travel times within a discrete-scenario uncertainty set. Thus, the robust counterpart formulation ensures that each constraint holds for all predefined scenarios. Moreover, the former study adopts a bi-objective approach, also aiming to minimise the travelled distance.

To the best of our knowledge, the incorporation of the RO paradigm in the PDPTW has not yet been fully explored. Our study addresses this gap by proposing methods for handling demand and travel time uncertainties in the PDPTW using RO. We are aware of only one study related to the RPDPTW, which focuses on a rich variant. Specifically, Liu et al. (2023) consider a homogeneous fleet of battery-powered electric vehicles in the PDPTW under demand uncertainty. They rely on a scenario-based uncertainty set constructed from the extreme points of a cardinality-constrained polytope. The authors propose a branch-and-cut-and-price algorithm, which is tested on instances with 20, 25, and 30 requests.

Table 1 presents a summary of the main aspects of papers that addressed the PDP

under uncertainty via RO. For each related paper, we have analysed which PDP variant it addresses (column *Variant*), the uncertain parameters considered (column *Uncertainty*), the set it is modelled in (column *Support*), and which method is used to obtain the robust formulation (column *Method*).

Paper	Variant	Uncertainty	Support	Method
One-to-many-to-one (1-M-1)				
Tajik et al. (2014)	1D	T,C,S,O	BB	DU
Mousavi and Vahdani (2017)	2E,1D	T,C,Q,O	BB	DU
Golsefidi and Jokar (2020)	SD	D,O	CC	DU
Santos et al. (2020)	1D,BH	O	CC,FM	DU
Mondal and Roy (2021)	SD	D,T,C,O	DS	NA
Vakili, Shirazi and Gitinavard (2021)	SD	D,T	DS	NA
Pavlova and Shichiyakh (2021)	SD	D,T,C,S,O	DS	NA
Pourmohammad-Zia et al. (2023)	1D,BH	K	EL	O
Guo et al. (2024)	SD	C	MC	O
Many-to-many (M-M)				
Chami et al. (2018)	DQ,SR	T	DS	NA
Huang et al. (2020)	SD	D,C	DS	NA
Allahyari, Yaghoubi and Woensel (2021)	SD	D	HW	NA
Hansuwa, Kumar and Chandrasekharan (2022)	SD	D,T,S	CC,EL	DU,LI
Chami et al. (2022)	DQ,SR	T	DS	NA
One-to-one (1-1)				
Liu et al. (2023)	EV	D	DS	NA

Notation description. SD: simultaneous pickup and delivery; 1D: single demand; BH: backhaul; EV: electric vehicles; DQ: depot with demand; SR: selective requests; 2E: two-echelon routing; D: demand; Q: capacity; T: travel time; S: service time; C: travel cost; K: fleet size; HW: hard worst-case scenario; DS: discrete scenarios uncertainty set; EL: ellipsoidal uncertainty set; CC: cardinality-constrained uncertainty set; BB: bounded box uncertainty set; KS: knapsack uncertainty set; FM: factor model uncertainty set; MC: multi-interval cardinality constrained uncertainty set. DU: dualisation scheme; LI: linearisation scheme; NA: not applicable; O: other.

Table 1 – Summary of publications on robust pickup and delivery problems.

Chapter 3
Problem description

We define the problem over a graph $G = (\mathcal{N}, \mathcal{A})$, where the set of nodes is given by $\mathcal{N} = \{0, 2n+1\} \cup \mathcal{P} \cup \mathcal{D}$. Nodes 0 and $2n+1$ represent the initial and final depot, respectively, although they correspond to the same physical location. The set $\mathcal{P} = \{1, \ldots, n\}$ represents the pickup nodes, while the set $\mathcal{D} = \{n+1, \ldots, 2n\}$ represents the delivery nodes. Let $\mathcal{R} = \{1, \ldots, n\}$ denote the set of requests, where each pickup node $i \in \mathcal{P}$ is paired with its corresponding delivery node $n+i \in \mathcal{D}$, ensuring a unique pickup and delivery pair for every request $i \in \mathcal{R}$. Moreover, let $\mathcal{C} = \mathcal{P} \cup \mathcal{D}$ denote the set of pickup and delivery nodes, hereafter referred to as the *customer node set*. The arc set is defined as $\mathcal{A} = \{(i,j) : i \in \mathcal{N} \setminus \{2n+1\}, j \in \mathcal{N} \setminus \{0, i\}, i \neq j+n\}$, representing all possible arcs between pairs of nodes. The set $\mathcal{A}$ can be reduced through an arc elimination procedure, which considers precedence, capacity, and time window constraints (Dumas; Desrosiers; Soumis, 1991; Ropke; Cordeau; Laporte, 2007; Ropke; Cordeau, 2009).

Each request $i \in \mathcal{R}$ has a demand o_i, which corresponds to a demand value for its pickup node, $q_i = o_i$, and its delivery node, $q_{i+n} = -o_i$. Each node $i \in \mathcal{N}$ has a service time d_i and a time window $[e_i, l_i]$. The time windows are treated as hard constraints: vehicles cannot arrive after the closing time, but they are allowed to wait if they arrive before the opening time. We assume that the depot has no demand or service time, i.e., $q_0 = q_{2n+1} = 0$ and $d_0 = d_{2n+1} = 0$, and its time window is sufficiently large to accommodate any route based on the customers' time windows. Each arc $(i,j) \in \mathcal{A}$ is associated with a travel cost c_{ij} and a travel time t_{ij}, both of which satisfy the triangle inequality. Finally, we consider a homogeneous fleet of vehicles, each with a capacity Q.

In the deterministic PDPTW, routes departing from and returning to the depot are planned for vehicles to serve a set of requests. Each request consists of transporting items from a pickup node to its paired delivery node. The routes must respect the time windows of the nodes and the capacity of the vehicles, considering that demands and travel times are precise, reliable, and known beforehand. The objective is to determine routes that minimise the total routing costs. In what follows, we detail the incorporation of uncertainties into the PDPTW and discuss the features that arise in the RPDPTW.

3.1 Robust PDPTW

The RPDPTW considers both the requests' demand and the arcs' travel time as uncertain. RO approaches do not require complete knowledge of the probabilistic

distribution of the uncertain parameters. Instead, they rely on estimates of their worst-case values, typically defined as intervals of possible values. These estimates are used to construct an uncertainty set containing the anticipated parameter realisations. Methods are then developed to ensure feasibility for all parameter values within the uncertainty set.

The estimated nominal and deviation values associated with the demand of a request i are $\bar{o}_i$ and $\hat{o}_i$, respectively. Similarly, the estimated nominal and deviation values for the travel time associated with an arc (i, j) are $\bar{t}_{ij}$ and $\hat{t}_{ij}$, respectively. Thus, the demand realisation of each request $i \in \mathcal{R}$ and the travel time realisation on each arc $(i, j) \in \mathcal{A}$ belong to the intervals $\tilde{o}_i \in [\bar{o}_i - \hat{o}_i, \bar{o}_i + \hat{o}_i]$ and $\tilde{t}_{ij} \in [\bar{t}_{ij} - \hat{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}]$, respectively. Let $\xi^q_i, \xi^t_{ij} \in [-1, 1]$ be the normalised scale deviations of demand and travel time, respectively. They indicate the level of deviation from their nominal values, i.e., $\xi^q_i = (\tilde{o}_i - \bar{o}_i)/\hat{o}_i$ and $\xi^t_{ij} = (\tilde{t}_{ij} - \bar{t}_{ij})/\hat{t}_{ij}$. The demand and travel time realisations can then be expressed as a linear combination of their nominal and deviation components: $\tilde{o}_i = \bar{o}_i + \xi^q_i \hat{o}_i$ and $\tilde{t}_{ij} = \bar{t}_{ij} + \xi^t_{ij} \hat{t}_{ij}$. The corresponding pickup and delivery demands of a request $i \in \mathcal{R}$ are given by $\tilde{q}_i = \tilde{o}_i$ and $\tilde{q}_{i+n} = -\tilde{o}_i$, respectively. Therefore, the scale deviation ξ^q_i of a request i ensures that the demand realisation at pickup i is paired with the demand realisation at its corresponding delivery $i + n$.

In our approach, the realisations of the uncertain demands and travel times belong to a polyhedral uncertainty set. Specifically, we consider the cardinality-constrained uncertainty set, which controls the level of conservatism using a predefined *budget of uncertainty* (Bertsimas; Sim, 2004). If a budget of uncertainty Γ is set as an integer value, the worst-case scenario happens when Γ uncertain parameter components attain their total deviation while the remaining components keep in their nominal value. O the other hand, if Γ has a fractional value, the worst-case scenario is the one by making $\lfloor \Gamma \rfloor$ parameter components attaining their total deviation and a single other component assuming the fraction $\Gamma - \lfloor \Gamma \rfloor$ of its deviation while the rest of components are in their nominal value (Bertsimas; Sim, 2004). In the considered problem, let $\Gamma^q, \Gamma^t \in \mathbb{N}_0$ denote the budgets of uncertainty for the demand and travel time uncertainty sets, respectively. These parameters limit the number of requests (in the case of demand) and arcs (in the case of travel time) that are allowed to simultaneously attain their worst-case values.

The worst-case realisation of a parameter occurs when it attains either its leftmost or rightmost extreme value within its interval. Given that a solution is feasible for the nominal parameter values, capacity or time windows would not be violated if demands or travel times were smaller. Instead, increasing these values could potentially result in a violation. Therefore, the domain of the normalised scale deviations can be restricted to $\xi^q, \xi^t \in [0, 1]$ without loss of generality. The cardinality-constrained uncertainty sets for

demand $\mathcal{U}^q$ and travel time $\mathcal{U}^t$ can be defined as follows:

$$\mathcal{U}^q = \left\{ \tilde{q} \in \mathbb{R}^{|\mathcal{C}|} : \tilde{q}_i = \bar{q}_i + \hat{q}_i \xi_i^q, i \in \mathcal{P}; \tilde{q}_i = \bar{q}_i + \hat{q}_i \xi_{i-n}^q, i \in \mathcal{D}; \sum_{i \in \mathcal{P}} \xi_i^q \leq \Gamma^q; \xi_i^q \in [0,1]^{|\mathcal{P}|} \right\}, \tag{3.1.1}$$

$$\mathcal{U}^t = \left\{ \tilde{t} \in \mathbb{R}_+^{|\mathcal{A}|} : \tilde{t}_{ij} = \bar{t}_{ij} + \xi_{ij}^t \hat{t}_{ij}, (i,j) \in \mathcal{A}; \sum_{(i,j) \in \mathcal{A}} \xi_{ij}^t \leq \Gamma^t; \xi_{ij}^t \in [0,1]^{|\mathcal{A}|} \right\}. \tag{3.1.2}$$

The deterministic case is obtained by setting $\Gamma^q = \Gamma^t = 0$, meaning no uncertainty is considered. On the other hand, the configuration $\Gamma^q = n$ and $\Gamma^t = |\mathcal{A}|$ represents the hard worst-case scenario, where all uncertain components simultaneously attain their worst-case values, resulting in an overly pessimistic situation.

A solution is said to be *robust feasible* if all the following conditions are satisfied: (i) each vehicle departs from and returns to the depot; (ii) each customer node is visited exactly once; (iii) the pickup and delivery nodes of a request are visited by the same vehicle; (iv) pickup nodes are served before their corresponding delivery nodes; (v) vehicle capacity is not exceeded for any demand realisation within the uncertainty set; and (vi) customers' time windows are not violated for any travel time realisation within the uncertainty set. This problem is NP-hard, as it generalises the VRPTW.

3.2 Recursive equations

RO approaches modelled using certain polyhedral uncertainty sets can leverage dynamic programming recursive equations to verify whether a solution is robust feasible (Agra et al., 2013; Munari et al., 2019; Campos; Coelho; Munari, 2024). In what follows, we demonstrate how to compute the worst-case vehicle load and arrival time in the RPDPTW.

Different from the calculation used in previously addressed VRP variants, the recursive equations for verifying capacity robustness under demand uncertainty must explicitly consider the pairing property inherent to the RPDPTW. Hence, to determine the vehicle load at delivery nodes, we introduce the sets $\mathcal{W}_{u_j\gamma}$, for $j = 0, \ldots, k$ and $\gamma = 0, \ldots, \Gamma^q$, which track the pickup nodes whose demand realisations deviate from their nominal values. These sets allow us to ensure that the same demand realisation applies to the paired pickup and delivery nodes of a request. Consider a route $u = (u_0, u_1, ..., u_k)$ with $u_0 = 0$ and $u_k = 2n + 1$. Let $Q_{u_j\gamma}$ represent the vehicle load at node u_j of the route, when the demand of up to γ requests attain their worst-case values, for $j = 0, \ldots, k$ and $\gamma = 0, \ldots, \Gamma^q$. Furthermore, let $\mathcal{W}_{u_j\gamma} \subseteq \mathcal{P}$ denote the set of all pickup nodes which their demand realisations have simultaneously attained their worst-case along the path up to node u_j, assuming that it happens for at most γ nodes. This set is formally defined as follows:

$$\mathcal{W}_{u_j\gamma} = \left\{ u_i \in \mathcal{P} : Q_{u_{i-1}\gamma-1} + \hat{q}_{u_i} > Q_{u_{i-1}\gamma}, i = 1, \ldots, j \right\},$$

$$j = 0, \ldots, k, \gamma = 1, \ldots, \Gamma^q. \quad (3.2.1)$$

Using this auxiliary set, we can calculate the worst-case vehicle load via the following recursive equation, for $j = 0, \ldots, k$ and $\gamma = 0, \ldots, \Gamma^q$:

$$Q_{u_j\gamma} = \begin{cases} 0, & \text{if } j = 0, & (3.2.2a) \\ Q_{u_{j-1}\gamma} + \bar{q}_{u_j}, & \text{if } u_j \in \mathcal{P} \wedge \gamma = 0, & (3.2.2b) \\ \max\{Q_{u_{j-1}\gamma} + \bar{q}_{u_j},\ Q_{u_{j-1},\gamma-1} + \bar{q}_{u_j} + \hat{q}_{u_j}\}, & \text{if } u_j \in \mathcal{P} \wedge \gamma > 0, & (3.2.2c) \\ Q_{u_{j-1},\gamma} + \bar{q}_{u_j}, & \text{if } u_j \in \mathcal{D} \wedge u_j - n \notin \mathcal{W}_{u_{j-1},\gamma}, & (3.2.2d) \\ Q_{u_{j-1},\gamma} + \bar{q}_{u_j} + \hat{q}_{u_j}, & \text{if } u_j \in \mathcal{D} \wedge u_j - n \in \mathcal{W}_{u_{j-1},\gamma}. & (3.2.2e) \end{cases}$$

Cases (3.2.2a)–(3.2.2c) are analogous to the recursive equations stated for the RVRPTW (Munari et al., 2019), but are applied exclusively to pickup nodes. Case (3.2.2a) represents the boundary condition, ensuring that there is no vehicle load after leaving the depot. In Case (3.2.2b), the vehicle load is incremented by the nominal demand of the visited pickup node if the demand of no request is assumed to deviate from its nominal value, i.e., $\gamma = 0$. On the contrary, Case (3.2.2c) ensures that the vehicle load is incremented either by the nominal demand alone or by both the nominal and deviation demands, i.e., its demand worst-case value. To determine whether the demand of a pickup node has reached its worst-case value, we verify if $Q_{u_{j-1}\gamma-1} + \hat{q}_{u_j} > Q_{u_{j-1}\gamma}$, as defined in Case (3.2.2c). Finally, the pairing property between the pickup and delivery nodes of a request is ensured by matching their demand realisations in cases (3.2.2d) and (3.2.2e).

In VRP variants with vehicle load that either increases or decreases monotonically along the route (e.g., with only pickups or only deliveries), the worst-case load of a route can be easily computed (Munari et al., 2019; Subramanyam; Repoussis; Gounaris, 2020). In this case, the worst-case load is determined by considering the Γ^q-largest demand deviations under the cardinality-constrained uncertainty set. However, this assumption does not hold in PDPs, since the vehicle load does not increase (decrease) monotonically. Yet, the pairing property allows for the computation of the worst-case load by considering only the *unpaired nodes*, depending on the order of visits. For a given set of customers $\mathcal{S} \subset \mathcal{C}$, a node $i \in \mathcal{S}$ is defined as an unpaired node if its corresponding pickup or delivery node is not in $\mathcal{S}$.

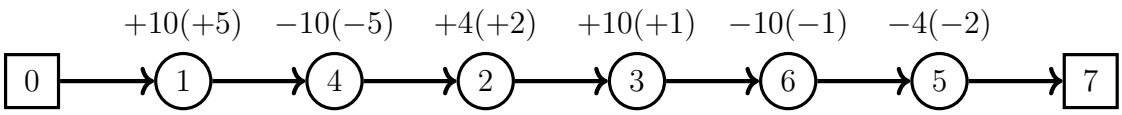
Proposition 1. *The worst-case vehicle load Q_h at node u_h of a feasible PDPTW route $(u_0, u_1, \ldots, u_k)$ with $1 \leq h \leq k-1$ is obtained by summing the nominal demands and the $\Gamma = \min\{\Gamma^q, h\}$ largest demand deviations of the visited unpaired pickup nodes up to u_h. Let $\mathcal{S}_h \subseteq \mathcal{C}$ be the set of customers up to position h in the route, and let $\mathcal{S}_h^+ = \{i \in \mathcal{S}_h \cap \mathcal{P} : i + n \notin \mathcal{S}_h\}$ denote the set of its unpaired nodes. Then, the worst-case load can be expressed as:*

$$Q_h = \sum_{i \in \mathcal{S}_h^+} \bar{q}_i + \max_{\substack{\mathcal{Q} \subseteq \mathcal{S}_h^+ \\ |\mathcal{Q}| \leq \Gamma}} \sum_{i \in \mathcal{Q}} \hat{q}_i \qquad 1 \leq h \leq k. \quad (3.2.3)$$

Proof. Let $\mathcal{V}_h = \mathcal{S}_h \setminus \mathcal{S}_h^+$ be the set of customer nodes where both the corresponding pickup and delivery nodes are included in the route up to position h. If any subset of paired nodes in $\mathcal{V}_h$ attains their worst-case demand values, the value Q_h remains unaffected, as the demand realisations of a pickup node and its corresponding delivery node are equal in absolute value but opposite in sign, hence, cancelling each other out. As a result, only the worst-case demand values of unpaired nodes (i.e., unpaired pickup nodes) contribute to the vehicle load Q_h.

□

For example, consider an instance with $\Gamma^q = 1$, vehicle capacity $Q = 15$, and three requests, with their demands belonging to the intervals $\tilde{o}_1 \in [10, 15]$, $\tilde{o}_2 \in [4, 6]$, and $\tilde{o}_2 \in [10, 11]$. Also, consider a route solution is given by $(0, 1, 4, 2, 3, 6, 5, 7)$. Figure 1 illustrates this example, showing each customer's nominal and deviation demand values above the nodes, along with the vehicle load after visiting the customers. Below the path, we illustrate how the vehicle load varies when the demand of each request (1, 2, and 3, respectively) attains its worst-case value. The deviation of request 2 determines the worst-case that makes the path infeasible, instead of the deviation of request 1, which is the largest. The demand of request 1 is delivered immediately after being collected. Then, request 2 has the larger demand worst-case value among the unpaired pickup nodes nodes up to node 3.



Worst-case of $\tilde{o}_1$:	15	0	4	14	4	0
Worst-case of $\tilde{o}_2$:	10	0	6	<u>16</u>	6	0
Worst-case of $\tilde{o}_3$:	10	0	4	15	4	0

Figure 1 – Illustration of the order-dependent characteristic of the demand worst-case.

Given that the time window constraints in the RPDPTW are similar to those in the RVRPTW, their robustness under travel time uncertainty is checked in the same way as in (Munari et al., 2019). Let $B_{u_j\gamma}$ denote the service start time at node u_j when the travel times of up to γ arcs attain their worst-case values, for $j = 0, \ldots, k$ and $\gamma = 0, \ldots, \Gamma^t$. The service start time can be calculated recursively as follows:

$$B_{u_j\gamma} = \begin{cases} e_{u_j}, & \text{if } j = 0, \quad (3.2.4a) \\ \max\{e_{u_j}, B_{u_{j-1}\gamma} + d_{u_{j-1}} + \bar{t}_{u_{j-1},u_j}\}, & \text{if } \gamma = 0, \quad (3.2.4b) \\ \max\{e_{u_j}, B_{u_{j-1},\gamma} + d_{u_{j-1}} + \bar{t}_{u_{j-1},u_j}, & \\ \qquad B_{u_{j-1},\gamma-1} + d_{u_{j-1}} + \bar{t}_{u_{j-1},u_j} + \hat{t}_{u_{j-1},u_j}\}, & \text{otherwise.} \quad (3.2.4c) \end{cases}$$

Case (3.2.4a) represents the boundary condition at the depot, stating that the service at the depot begins at its time window's opening time. Case (3.2.4b) ensures that

if the travel time of no arc attains its worst-case value, $\gamma = 0$, at node u_j, the service start time is the maximum between the time window's opening time of u_j and the end of the service at u_{j-1} plus the nominal travel time $\bar{t}_{u_{j-1},u_j}$. Case (3.2.4c) accounts for the scenario where the service start time is the maximum between the time window's opening time, the elapsed time with the nominal travel time, and the elapsed time with the worst-case travel time value. For the time window constraints to be robustly feasible, it suffices to verify the time window constraint at customer u_j under the assumption that the travel time of up to Γ^t arcs attain their worst-case value, i.e., $B_{u_j\Gamma^t} \leq l_{u_j}$.

Chapter 4
Methods for the RPDPTW

In this study, the formulation proposed by Furtado, Munari and Morabito (2017) is used as the basis to derive robust counterpart formulations for the RPDPTW. This formulation introduces continuous variables in the two-index vehicle flow model to ensure that the pickup and delivery nodes of a request are served by the same vehicle. The choice of this model is motivated by its good performance with general-purpose mixed-integer programming (MIP) solvers compared to other compact formulations for the (deterministic) PDPTW.

Additionally, consider $x_{ij} \in \{0,1\}$ as a binary variable that takes the value 1 if node j is visited immediately after node i by the same vehicle, i.e., the arc $(i,j) \in \mathcal{A}$ is travelled, and 0 otherwise. The vehicle load after serving node $i \in \mathcal{N}$ is represented by the continuous variable $Q_i \in \mathbb{R}_+$. The service start time at node $i \in \mathcal{N}$ is represented by the continuous variable $B_i \in \mathbb{R}_+$. The variables $v_i \in \mathbb{R}_+$ denote the index of the first visited customer on the path from the depot to node $i \in \mathcal{C}$. These variables are used to ensure the pairing of pickup and delivery nodes, i.e., both nodes must be visited by the same vehicle. The formulation for the (deterministic) PDPTW is as follows:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{4.0.1}$$

$$\text{s.t.} \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 1, \qquad j \in \mathcal{C} \tag{4.0.2}$$

$$Q_j \geq Q_i + q_j - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} \tag{4.0.3}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\}, \qquad i \in \mathcal{N} \tag{4.0.4}$$

$$B_j \geq B_i + t_{ij} + d_i - (l_i - e_j)(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} \tag{4.0.5}$$

$$e_i \leq B_i \leq l_i, \qquad i \in \mathcal{N} \tag{4.0.6}$$

$$B_{n+i} \geq B_i + t_{i,n+i}, \qquad i \in \mathcal{P} \tag{4.0.7}$$

$$v_{n+i} = v_i, \qquad i \in \mathcal{P} \tag{4.0.8}$$

$$jx_{0j} \leq v_j \leq jx_{0j} - n(x_{0j} - 1), \qquad j \in \mathcal{C} \tag{4.0.9}$$

$$n(x_{ij} - 1) \leq v_j - v_i \leq n(1 - x_{ij}), \qquad i,j \in \mathcal{C} : i \neq j \tag{4.0.10}$$

$$1 \leq v_j \leq n, \qquad j \in \mathcal{C} \tag{4.0.11}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in \mathcal{A}. \tag{4.0.12}$$

The objective function (4.0.1) seeks to minimise the total routing costs. Constraints (4.0.2) ensure that each customer node is visited exactly once and that the vehicle flow is consistent throughout the network. Constraints (4.0.3) propagate the load on the vehicle

travelling between two nodes, while constraints (4.0.4) ensure that the vehicle's capacity is not exceeded. Constraints (4.0.5) and (4.0.6) propagate the time flow, prevent subtours, and ensure that time windows are respected. Constraints (4.0.7)–(4.0.10) guarantee the precedence and pairing relationships between the pickup and delivery nodes of a request. The domains of the variables are defined in (4.0.4), (4.0.6), (4.0.11), and (4.0.12).

From formulation (4.0.1)–(4.0.12), we derive two robust counterpart formulations with different approaches to ensure the vehicle's capacity and the nodes' time windows for any demand and travel time realisations belonging to the uncertainty sets (3.1.1) and (3.1.2). Our first approach, detailed in Section 4.1, employs the linearisation technique to produce a compact formulation. The second approach, described in Section 4.2, models capacity and time window robustness by incorporating cutting planes into the problem using a B&C algorithm. While the advantage of the compact formulation lies in its ease of implementation with general-purpose MIP solvers, B&C algorithms typically achieve better performance on large-sized instances.

4.1 Compact formulation

Tractable robust counterpart formulations can also be derived using the dualisation technique for polytope-based uncertainty sets. While this technique has been applied to RVRP variants, it often requires additional constraints, variables, or intricate reformulations (Agra et al., 2012; Gounaris; Wiesemann; Floudas, 2013; De La Vega; Munari; Morabito, 2019). In contrast, the linearisation scheme, which integrates recursive equations into compact vehicle flow formulations, has demonstrated superior performance on general-purpose MIP solvers (Munari et al., 2019; Campos; Coelho; Munari, 2024). We show how to derive robust counterpart compact formulations for the RPDPTW using the dualisation approach in Appendix A.

In the linearisation scheme, recursive equations are explicitly incorporated into the formulation to ensure solution robustness. This approach considers the structure of the uncertainty set, along with the definition of the recursive equations, to extend the deterministic model to its robust counterpart. The degree constraints, pairing constraints, and the domains of their related variables remain the same as in the deterministic formulation.

Let $Q_{j\gamma} \in \mathbb{R}_+$ be the vehicle load after serving node $j \in \mathcal{N}$, considering that the demand of γ requests simultaneously attain their worst-case values. Note that these variables have the same meaning as in the recursive equation (3.2.2). Let $W_{j\gamma h}$ be a binary variable that assumes the value 1 if, and only if, the demand of pickup node $h \in \mathcal{P}$ has attained its worst-case on the path from the depot to node $j \in \mathcal{N}$, while γ requests are simultaneously considered in their worst-case demand values. The variable $W_{j\gamma h}$ is related to set $\mathcal{W}_{j\gamma}$ defined in Subsection 3.2, and thus is used to identify the requests attaining

their worst-case demand values, as defined in (3.2.1), within the formulation. Thus, the recursive equation (3.2.2) can be reformulated as the following set of linear constraints:

$$\max\{0, \bar{q}_j\} \leq Q_{j\gamma} \leq \min\{Q, Q + \bar{q}_j\}, \qquad j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^q \tag{4.1.1}$$

$$Q_{j\gamma} \geq Q_{i\gamma} + \bar{q}_j x_{ij} - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 0, \ldots, \Gamma^q \tag{4.1.2}$$

$$Q_{j\gamma} \geq Q_{i,\gamma-1} + (\bar{q}_j + \hat{q}_j) x_{ij} - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 1, \ldots, \Gamma^q \tag{4.1.3}$$

$$Q_{j\gamma} \geq Q_{i\gamma} + \bar{q}_j x_{ij} + \hat{q}_j W_{i,\gamma,j-n} - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} : j \in \mathcal{D}, \gamma = 0, \ldots, \Gamma^q. \tag{4.1.4}$$

The domain of $Q_{j\gamma}$ is defined by constraints (4.1.1), which ensure that the vehicle capacity is respected at each node for any level of uncertainty. Constraints (4.1.2) and (4.1.3) jointly correspond to cases (3.2.2b) and (3.2.2c), ensuring that $Q_{j\gamma}$ takes the maximum value when the demand of pickup node j either attains or does not attain its worst-case value. Specifically, constraints (4.1.2) ensure that the load $Q_{j\gamma}$ increases by at least the nominal demand of j, while constraints (4.1.3) also account for the customer demand deviation when the number of requests in their worst-case (γ) increases by one unit. Constraints (4.1.4) ensure that the demand realisation of the delivery node j matches the demand realisation of the corresponding pickup node $j - n$, and that the vehicle load is correctly updated after a delivery.

The appropriate value of each variable $W_{j\gamma h}$ in the formulation can be ensured using a set of linear constraints. Let us assume that an arc $(i,j) \in \mathcal{A}$, where $j \in \mathcal{P}$, is travelled ($x_{ij} = 1$). Recall from set definition (3.2.1) that the demand of the pickup node j attains its worst-case if $Q_{i,\gamma-1} + \hat{q}_j - Q_{i\gamma} > 0$. With a few algebraic operations, one can derive that $(Q_{i,\gamma-1} + \hat{q}_j - Q_{i\gamma})/(\hat{q}_j + \max_{i \in \mathcal{P}} \hat{q}_i) \in [0,1]$. This expression indicates whether the demand of node j attains its worst-case, which occurs when the result is greater than 0. We can therefore use the following set of linear constraints:

$$W_{j\gamma j} \geq \frac{Q_{i,\gamma-1} + \hat{q}_j - Q_{i\gamma}}{\hat{q}_j + \max_{i \in \mathcal{P}} \hat{q}_i} + x_{ij} - 1, \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 1, \ldots, \Gamma^q \tag{4.1.5}$$

$$W_{j\gamma j} \geq W_{i,\gamma-1,j} + \frac{Q_{i,\gamma-1} + \hat{q}_j - Q_{i\gamma}}{\hat{q}_j + \max_{i \in \mathcal{P}} \hat{q}_i} + x_{ij} - 1, \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 1, \ldots, \Gamma^q \tag{4.1.6}$$

$$W_{j\gamma h} \geq W_{i\gamma h} + x_{ij} - 1, \qquad (i,j) \in \mathcal{A}, h \in \mathcal{P}, \gamma = 0, \ldots, \Gamma^q \tag{4.1.7}$$

$$W_{j\gamma h} \in \{0,1\}, \qquad j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^q, h \in \mathcal{P}. \tag{4.1.8}$$

Constraints (4.1.5) ensure that $W_{j\gamma j} = 1$ if the demand of the pickup node j attains its worst-case. Constraints (4.1.6) ensure that $W_{i,\gamma-1,j} = 0$ if the demand of the pickup node j attains its worst-case and $x_{ij} = 1$. Constraints (4.1.7) guarantee consistency along the route if the demand of the pickup node h attains its worst-case value at any point on the path from the depot to node j. The domain of the variables is defined in (4.1.8). It is worth mentioning that, to the best of our knowledge, the variables $W_{j\gamma h}$ and the linearized constraints (4.1.1)–(4.1.8) are introduced in this study, as they are specifically developed to account for paired pickup and delivery operations.

The linearisation of the recursive equation (3.2.4), which calculates the worst-case service start time under uncertain travel times, is performed as described in Munari et al. (2019). Let $B_{j\gamma}$ denote the service start time at node $j \in \mathcal{N}$, considering that the travel times of up to γ arcs simultaneously attain their worst-case values. The robust counterpart compact formulation, which incorporates demand and travel time uncertainties in the PDPTW, is defined as follows:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{4.1.9}$$

$$\text{s.t.} \quad (4.0.2), (4.0.8) - (4.1.8)$$

$$B_{j\gamma} \geq B_{i\gamma} + d_i + \bar{t}_{ij} - (l_i - e_j)(1 - x_{ij}), \qquad (i,j) \in \mathcal{A}, \gamma = 0, \ldots, \Gamma^t \tag{4.1.10}$$

$$B_{j\gamma} \geq B_{i,\gamma-1} + d_i + \bar{t}_{ij} + \hat{t}_{ij} - (l_i - e_j)(1 - x_{ij}), \qquad (i,j) \in \mathcal{A}, \gamma = 1, \ldots, \Gamma^t \tag{4.1.11}$$

$$B_{n+j,\gamma} \geq B_{j\gamma} + d_j + \bar{t}_{j,n+j}, \qquad j \in \mathcal{P}, \gamma = 0, \ldots, \Gamma^t \tag{4.1.12}$$

$$e_j \leq B_{j\gamma} \leq l_j, \qquad j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^t. \tag{4.1.13}$$

The objective function (4.1.9) seeks to minimise the total routing cost. Constraints (4.0.2) (degree constraints), (4.0.8)–(4.0.10) (pairing constraints), and (4.0.11) and (4.0.12) (variable domains) are similar to those in the deterministic case. Capacity robustness under demand uncertainty is ensured by constraints (4.1.1)–(4.1.8). Time window robustness under travel time uncertainty is ensured by constraints (4.1.10)–(4.1.13). Constraints (4.1.10) and (4.1.11) function as cases (3.2.4a) and (3.2.4b), ensuring that the service start time is, or is not, the maximum value when the travel time in arc $(i,j) \in \mathcal{A}$ attains, or does not attain, its worst-case value. The precedence relationship between the pickup and delivery nodes of a request is enforced by constraints (4.1.12). Constraints (4.1.13) ensure that time windows are not violated for any level of uncertainty.

4.2 Cutting-plane formulation

The formulation is initialised with the deterministic formulation (4.0.2)–(4.0.12), including the degree constraints, capacity constraints, time window constraints, precedence constraints, and pairing constraints, along with their associated variable domains. In particular, the deterministic capacity constraints (4.0.3) and (4.0.4), and time window constraints (4.0.5) and (4.0.6), are incorporated using the nominal values of demands and travel times. In what follows, we derive the exponential-sized constraints based on the notation introduced in Section 3.1.

Vehicle capacity is often ensured using rounded capacity inequalities (RCIs) in both deterministic VRP variants and their robust counterparts (Gounaris; Wiesemann; Floudas, 2013; Campos; Coelho; Munari, 2024). The RCI provides a lower bound on the number of vehicles required to serve a set of customers $\mathcal{S} \subseteq \mathcal{C}$, considering their total demand. A general form of RCIs is presented in Equation (4.2.1), where $r(\mathcal{S})$ represents the total

demand in $\mathcal{S}$. Let $x(\delta^+(\mathcal{S})) = \sum_{(i,j)\in\delta^+(\mathcal{S})} x_{ij}$ denote the summation of x_{ij} over the set of arcs connecting nodes within and outside $\mathcal{S}$, where $\delta^+(\mathcal{S}) = \{(i,j) \in \mathcal{A} : i \notin \mathcal{S}, j \in \mathcal{S}\}$. Note that maximisation is required because $\mathcal{S}$ could contain only paired nodes.

$$x(\delta^+(\mathcal{S})) \geq \max\left\{1, \left\lceil \frac{r(\mathcal{S})}{Q} \right\rceil\right\}, \qquad \mathcal{S} \subseteq \mathcal{C} : |\mathcal{S}| \geq 2. \tag{4.2.1}$$

The computation of $r(\mathcal{S})$, considering that the demand realisations belong to a cardinality-constrained uncertainty set, can be incorporated based on a generalisation of Proposition 1. We propose the *robust RCI* (RRCI) for the RPDPTW under demand uncertainty, taking into account the total worst-case demand of either unpaired pickup nodes or unpaired delivery nodes. For any set of customers $\mathcal{S} \subseteq \mathcal{C}$, let $\mathcal{S}^+ = \{i \in \mathcal{S} \cap \mathcal{P} : i + n \notin \mathcal{S}\}$ be the set of unpaired pickup nodes, and let $\mathcal{S}^- = \{i \in \mathcal{S} \cap \mathcal{D} : i - n \notin \mathcal{S}\}$ denote the set of unpaired delivery nodes. The value of $r(\mathcal{S})$ can then be given as follows:

$$r(\mathcal{S}) = \max\left\{\sum_{i\in\mathcal{S}^+} \bar{q}_i + \max_{\substack{\mathcal{Q}\subseteq\mathcal{S}^+ \\ |\mathcal{Q}|\leq\Gamma^q}} \sum_{i\in\mathcal{Q}} \hat{q}_i, \left|\sum_{i\in\mathcal{S}^-} \bar{q}_i + \min_{\substack{\mathcal{Q}\subseteq\mathcal{S}^- \\ |\mathcal{Q}|\leq\Gamma^q}} \sum_{i\in\mathcal{Q}} \hat{q}_i,\right|\right\}. \tag{4.2.2}$$

Under travel time uncertainty, infeasible paths with respect to time windows can be eliminated using the classic *no-good* constraints by checking robustness in the separation procedure (Agra et al., 2013; Campos; Coelho; Munari, 2024). We implemented a strengthened version of this inequality, the tournament constraints of Ascheuer, Fischetti and Grötschel (2000), as also done in other robust routing variants (Wang; Subramanyam; Gounaris, 2022). We refer to these inequalities as robust infeasible path elimination constraints (RIPEC) hereafter.

Let $\mathcal{I}$ be the set of *minimal infeasible paths* with respect to time windows. An infeasible path $u = (u_1, u_2, \ldots, u_{k-1}, u_k)$ is said to be minimal if its subpaths $(u_2, \ldots, u_k)$ and $(u_1, \ldots, u_{k-1})$ are feasible (Kallehauge; Boland; Madsen, 2007). We define a set $\mathcal{L}_i(u)$, which contains the nodes to be considered in the RIPECs. For any infeasible path $u = (u_1, \ldots, u_k) \in \mathcal{I}$, let $\mathcal{L}_i(u)$ be defined as follows: $\mathcal{L}_i(u) = \{i+1\}$ if $u_i = 0, \{i+1, \ldots, k-1\}$ if $u_k = 2n+1 \wedge i \leq k-2$, and $\{i+1, \ldots, k\}$ otherwise. If the path includes the depot, we consider only its incoming or outgoing arcs. Furthermore, let $\bar{q}(\mathcal{S}) = \sum_{i\in\mathcal{S}} \bar{q}_i$ denote the total nominal demand, and let $\hat{q}(\mathcal{S}) = \sum_{i\in\mathcal{S}} \hat{q}_i$ denote the total demand deviation. The cutting-plane formulation can be presented as follows:

$$\min \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{4.2.3}$$

s.t. (4.0.2) – (4.0.12),

$$x(\delta^+(\mathcal{S})) \geq \max\left\{1, \left\lceil \frac{1}{Q}\left(\bar{q}(\mathcal{S}^+) + \max_{\substack{\mathcal{Q}\subseteq\mathcal{S}^+ \\ |\mathcal{Q}|\leq\Gamma^q}} \hat{q}(\mathcal{Q})\right)\right\rceil, \left\lceil\left|\frac{1}{Q}\left(\bar{q}(\mathcal{S}^-) + \min_{\substack{\mathcal{Q}\subseteq\mathcal{S}^- \\ |\mathcal{Q}|\leq\Gamma^q}} \hat{q}(\mathcal{Q})\right)\right|\right\rceil\right\},$$
$$\mathcal{S} \subseteq \mathcal{C} : |\mathcal{S}| \geq 2 \tag{4.2.4}$$

$$\sum_{i=1}^{k-1} \sum_{j \in \mathcal{L}_i(u)} x_{u_i,u_j} \leq k-2, \qquad u \in \mathcal{I}. \quad (4.2.5)$$

This formulation seeks to minimise the total routing cost (4.2.3) while satisfying the degree constraints (4.0.2), the capacity constraints (4.0.3)–(4.0.4), the time window constraints (4.0.5)–(4.0.6), the precedence constraints (4.0.7), and the pairing constraints (4.0.8)–(4.0.10), along with the variable domains (4.0.11)–(4.0.12). Furthermore, constraints (4.2.4) correspond to the RRCIs, which ensure the required number of vehicles to serve the set $\mathcal{S}$, considering the worst-case demand realisations of the unpaired customers within the set. In this way, the minimum number of vehicles needed to serve a set of nodes is defined based on the load either to drop off at the delivery nodes or to collect at the pickup nodes. Constraints (4.2.5) are the RIPECs, which ensure the elimination of infeasible paths. They restrict the number of arcs used in an infeasible path to be one less than the current number of arcs in the path, thereby forbidding the path. By considering the minimal infeasible paths in these constraints, all paths that contain a forbidden minimal path are also avoided. In Section 4.3, we describe the tailored B&C algorithm based on this formulation that we design, along with the corresponding separation procedures for the valid inequalities that we consider.

4.3 Branch-and-cut algorithm

The B&C algorithm dynamically incorporates RRCIs (4.2.4) and RIPECs (4.2.5) into the cutting-plane formulation. To identify violated constraints during the solution process, we developed two tailored separation algorithms, which are called for candidate incumbent (integer) and fractional solutions identified in the B&C tree. The separation of cuts for fractional solutions is restricted to the root node. In each execution of the separation algorithm, up to the 50 most violated cuts are added to the formulation. In each execution of the separation algorithm, we add up to the 50 most violated cuts are added to the formulation. The details of the separation algorithms are presented below.

4.3.1 Robust rounded capacity inequalities

The algorithm for separating RRCIs (4.2.4) consists of building sets of customers $\mathcal{S}$ for which the worst-case total demand yields a violated constraint. The algorithm runs iteratively, building customer sets starting from different initial customers. It begins at a given customer node and adds to $\mathcal{S}$ the customer node with the maximum incoming flow among all possible candidates. The worst-case total demand is updated by considering the demand deviation associated with the unpaired customers added to $\mathcal{S}$. When a violated RRCI is identified, it is added to the pool of inequalities, and the algorithm continues with the next customer as the initial node. After iterating over all customers as initial

nodes, up to the 50 most violated inequalities are incorporated into the formulation. This procedure is detailed in Algorithm 1, and Table 2 provides descriptions of the structures used in the algorithm.

Notation	**Description**
$\mathcal{S} \subseteq \mathcal{C}$	Set of customer nodes
$\mathcal{S}^+ \subseteq \mathcal{S}$	Set of unpaired pickup nodes
$\mathcal{S}^- \subseteq \mathcal{S}$	Set of unpaired delivery nodes
$\mathcal{Q}^+ \subseteq \mathcal{S}^+$	Set of unpaired pickup nodes with demands attaining their worst-case values
$\mathcal{Q}^- \subseteq \mathcal{S}^-$	Set of unpaired delivery nodes with demands attaining their worst-case values
$Q^+ \geq 0$	Total worst-case demand of the unpaired pickup nodes
$Q^- \leq 0$	Total worst-case demand of the unpaired delivery nodes

Table 2 – Description of the algorithm's notation for separating RRCIs.

In Algorithm 1, each node is considered in turn as the initial customer in $\mathcal{S}$, and the sets $\mathcal{S}$, $\mathcal{S}^+$, $\mathcal{S}^-$, $\mathcal{Q}^+$, and $\mathcal{Q}^-$ are initialised in Line 3. Variables Q^+ and Q^-, initialised in Line 4, quantify the total worst-case demand of the unpaired nodes in $\mathcal{S}^+$ and $\mathcal{S}^-$, respectively. Specifically, Q^+ represents the total demand collected at pickup nodes in the set and delivered to nodes outside of $\mathcal{S}$ (hereafter referred to as the *leaving load*). In contrast, Q^- represents the total demand delivered at delivery nodes in the set and originating from pickup nodes outside of $\mathcal{S}$ (hereafter referred to as the *entering load*). The worst-case leaving load Q^+ accounts for the nominal demand of all pickup nodes in $\mathcal{S}^+$ and the demand deviation of all pickup nodes in $\mathcal{Q}^+$. Similarly, the worst-case entering load Q^- and sets $\mathcal{S}^-$ and $\mathcal{Q}^-$ are defined for the unpaired delivery nodes.

At each node insertion into $\mathcal{S}$, the worst-case loads Q^- and Q^+ are updated in Lines 8–35. Specifically, the worst-case leaving load Q^+ is updated either in Lines 11–16 or in Lines 33–35. Similarly, the worst-case entering load Q^- is updated either in Lines 19–21 or in Lines 25–30. When a pickup node is inserted, if its corresponding delivery node is not in $\mathcal{S}$, the leaving load Q^+ is increased by the value of its demand realisation. Otherwise, the entering load Q^- is updated by removing the demand realisation associated with its corresponding delivery node. The cardinality of $\mathcal{Q}^+$ is limited by the budget of uncertainty Γ^q. If the cardinality of $\mathcal{Q}^+$ exceeds this threshold, the pickup node with the smallest demand deviation is discarded, and its associated demand deviation value is subtracted from Q^+. A similar procedure is applied when inserting a delivery node. If its corresponding pickup node is not in $\mathcal{S}$, its demand realisation is incorporated into the entering load Q^-. If the cardinality of $\mathcal{Q}^-$ reaches its maximum size, the delivery node with the lowest absolute demand value is discarded. Otherwise, if the corresponding pickup node is in $\mathcal{S}$, its demand realisation is disregarded, and $\mathcal{Q}^+$ is updated. If the set $\mathcal{S}$ results in a violated constraint, considering both Q^+ and Q^-, the corresponding RRCI is added to the list of violated cuts in Line 36, and a new set $\mathcal{S}$ is initialised with another customer

as the initial node. Otherwise, the process continues by selecting the next customer node j, not yet selected, that is associated with the maximum leaving flow from the set $\mathcal{S}$.

4.3.2 Robust infeasible path elimination constraints

The algorithm for separating RIPECs analyses paths associated with a given partial solution (integer or fractional) found during the B&C algorithm. The procedure to identify minimal infeasible paths for constraints (4.2.5) starts with each customer node $i \in \mathcal{C}$ and expands a path by selecting the node with the maximum flow leaving the previously added node. The path expansion continues while the path remains feasible with respect to time windows or while a violation of the corresponding RIPEC is still possible. We use the recursive equation (3.2.4) to calculate the worst-case service start time at each node in the path. Let $B_{j\gamma}$ represent the earliest service start time at node j when the travel time of up to γ arcs have deviated from their nominal values. In addition, we incorporate a subroutine using the set $\mathcal{E}$ to avoid redundancy in the algorithm for separating integer solutions. The set contains all nodes that can be skipped from being the initial node, in the main for-loop. The detailed procedure is provided in Algorithm 2, and Table 3 describes the structures used in the algorithm.

For each customer h as the initial node, we first initialise the path u and the service start time in Lines 7–13. During initialisation, if the initial customer node h is such that $x_{0h} > 0$, the path is initialised with the depot and the customer node, $u = (0, h)$, and the service start time is defined based on the travel time between 0 and h. Otherwise, the path contains only the customer node h, $u = (h)$, and the service start time is set to the opening of its time window e_h. This ensures that time window constraints are also considered for paths starting at the depot. Since individual arcs are robustly feasible due to our preprocessing stage there is no need to evaluate time window violations at the first two nodes in the path. The variable k represents the size of the path and is later used in a subroutine to identify a minimal infeasible path. In Lines 16–33, the path is expanded, and checks for time window and RIPEC violations are performed. We append to the path the node with the maximum flow from the previously added node. If the path results in an unviolated RIPEC, we stop expanding it in Line 21 and move on to the next initial node, as further expansion cannot lead to a violation. The worst-case service start time is calculated using the recursive equation (3.2.4) in Line 23. If a time window violation is detected (Line 24), we track the path backwards to identify the position r where the infeasibility originated. An infeasibility is deemed to originate at position r if the difference θ between the arrival time at the corresponding node and the opening of its time window (Line 28) is smaller than the violation size λ (Line 26), which is the difference between the arrival time at the last node and the closure of its time window. Finally, a RIPEC is added for the path $(u_r, \ldots, u_k)$ (Line 30). Since the RIPEC is incorporated considering the minimal infeasible path, using different initial nodes may result in the same constraint

Algorithm 1: Separation of RRCIs

function `SeparationRRCI(`x`)`
  foreach $h \in \mathcal{C}$ **do**
    $\mathcal{S}, \mathcal{S}^+, \mathcal{S}^-, \mathcal{Q}^+, \mathcal{Q}^- \leftarrow \emptyset$
    $Q^+, Q^- \leftarrow 0$
    $j \leftarrow h$
    do
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
      if $j \in \mathcal{P}$ **then**
        if $j + n \notin \mathcal{S}$ **then**
          $\mathcal{S}^+ \leftarrow \mathcal{S}^+ \cup \{j\}$
          $Q^+ \leftarrow Q^+ + \bar{q}_j$
          if $|\mathcal{Q}^+| < \Gamma^q$ **then**
            Update $Q^+ \leftarrow Q^+ + \hat{q}_j$ and $\mathcal{Q}^+ \leftarrow \mathcal{Q}^+ \cup \{j\}$
          else if $\hat{q}_j > \min_{i \in \mathcal{Q}^+}\{\hat{q}_i\}$ **then**
            $i \leftarrow \arg\min_{i \in \mathcal{Q}^+}\{\hat{q}_i\}$
            Update $Q^+ \leftarrow Q^+ - \hat{q}_i + \hat{q}_j$ and $\mathcal{Q}^+ \leftarrow \{\mathcal{Q}^+ \setminus \{i\}\} \cup \{j\}$
        else
          $\mathcal{S}^- \leftarrow \mathcal{S}^- \setminus \{j + n\}$
          $Q^- \leftarrow Q^- + \bar{q}_j$
          if $j + n \in \mathcal{Q}^-$ **then**
            Update $Q^- \leftarrow Q^- + \hat{q}_j$ and $\mathcal{Q}^- \leftarrow \mathcal{Q}^- \setminus \{j + n\}$
      else
        if $j - n \notin \mathcal{S}$ **then**
          $\mathcal{S}^- \leftarrow \mathcal{S}^- \cup \{j\}$
          $Q^- \leftarrow Q^- + \bar{q}_j$
          if $|\mathcal{Q}^-| < \Gamma^q$ **then**
            Update $Q^- \leftarrow Q^- + \hat{q}_j$ and $\mathcal{Q}^- \leftarrow \mathcal{Q}^- \cup \{j\}$
          else if $\hat{q}_j < \max_{i \in \mathcal{Q}^-}\{\hat{q}_i\}$ **then**
            $i \leftarrow \arg\max_{i \in \mathcal{Q}^-}\{\hat{q}_i\}$
            Update $Q^- \leftarrow Q^- - \hat{q}_i + \hat{q}_j$ and $\mathcal{Q}^- \leftarrow \{\mathcal{Q}^- \setminus \{i\}\} \cup \{j\}$
        else
          $\mathcal{S}^+ \leftarrow \mathcal{S}^+ \setminus \{j - n\}$
          $Q^+ \leftarrow Q^+ + \bar{q}_j$
          if $j - n \in \mathcal{Q}^+$ **then**
            Update $Q^+ \leftarrow Q^+ + \hat{q}_j$ and $\mathcal{Q}^+ \leftarrow \mathcal{Q}^+ \setminus \{j - n\}$
      if $x(\delta^+(\mathcal{S})) < \max\{1, \lceil \max\{Q^+, |Q^-|\}/Q \rceil\}$ **then**
        Add RRCI($\mathcal{S}$) to the pool of inequalities
        Break.
      $j \leftarrow \arg\max_{k \in \mathcal{C} \setminus \mathcal{S}}\{\sum_{i \in \mathcal{S}} x_{ik}\}$
    while $\exists k \in \mathcal{C} \setminus \mathcal{S} : \sum_{i \in \mathcal{S}} x_{ik} > 0$
  return pool of inequalities

Notation	Description
u	Path
$B_{j\gamma} \geq 0$	Service start time at node j with γ arcs in their travel time worst-case
$\mathcal{E} \subseteq \mathcal{N}$	Set of nodes already evaluated as the path's initial node
$k \geq 0$	Size of the path
$\lambda \geq 0$	Time windows violation size
$\theta \geq 0$	Service start time offset

Table 3 – Description of the algorithm's notation for separating RIPECs.

in integer solutions. This, however, does not hold for fractional solutions, as the maximum outgoing flow of a node can lead to an already visited node. As a result, different initial nodes may produce different paths. We use the set $\mathcal{E}$ to store all nodes that are not needed to be used as initial nodes. To avoid redundancy, once an infeasible path $(u_1, \ldots, u_k)$ is found and the RIPEC is added using the minimal infeasible path $(u_r, \ldots, u_k)$, nodes u_1 to u_r are excluded from further use as initial nodes, i.e., included in $\mathcal{E}$ (Line 31). Furthermore, if the entire path is feasible with respect to time windows, nodes u_1 to u_k are excluded from initiating further tentative paths in subsequent iterations (Line 34).

4.4 Preprocessing procedures

A preprocessing stage is carried out before applying the methods. This preprocessing involves a graph reduction through arc elimination and the incorporation of an initial pool of inequalities (Cordeau, 2006; Ropke; Cordeau; Laporte, 2007).

4.4.1 Arc elimination

Since the pickup node must be served before the delivery node of a given request, we eliminate all arcs $(0, i)$ and $(i - n, 2n + 1)$ for all $i \in \mathcal{D}$. Also, demand and travel time deviations are considered for reducing the graph. For uncertain demands, we remove from $\mathcal{A}$ the arcs $\{(i, j), (j, i), (i, j + n), (j, i + n), (i + n, j + n), (j + n, i + n)\}$, where $i, j \in \mathcal{P}$, if $\bar{q}_i + \bar{q}_j + \max\{\hat{q}_i, \hat{q}_j\} > Q$. For uncertain travel times, we remove from $\mathcal{A}$ the arcs (i, j) that satisfy $e_i + \bar{t}_{ij} + \hat{t}_{ij} > l_j$.

We also consider both demand and travel time uncertainties jointly. Consider the following arc elimination for any pair of pickup nodes $i, j \in \mathcal{P}$. The arcs $(i, n + j)$ and $(n + i, j)$ are removed from $\mathcal{A}$ if the paths $(j, i, n + j, n + i)$ and $(i, n + i, j, n + j)$ are infeasible, respectively. The arc (i, j) is removed from $\mathcal{A}$ if both paths $(i, j, n + i, n + j)$ and $(i, j, n + j, n + i)$ are infeasible. The arc $(n + i, n + j)$ is removed from $\mathcal{A}$ if both paths $(i, j, n + i, n + j)$ and $(j, i, n + i, n + j)$ are infeasible. Finally, the arcs $\{(i, j), (j, i), (n + i, j), (j, n + i), (n + j, i), (i, n + j), (n + i, n + j), (n + j, n + i)\}$ are removed from $\mathcal{A}$ if all

Algorithm 2: Separation of RIPECs

function `SeparationRIPEC(`x`)`
  $B_{j\gamma} \leftarrow 0$, for all $j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^t$
  $\mathcal{E} \leftarrow \emptyset$
  foreach $h \in \mathcal{C}$ **do**
    if $h \in \mathcal{E}$ **then**
      Continue to next h
    if $x_{0h} > 0$ **then**
      Set $u = (0, h)$ and $k \leftarrow 2$
      $B_{h0} \leftarrow \max\{e_h, e_0 + \bar{t}_{0h}\}$
      $B_{h\gamma} \leftarrow \max\{e_h, e_0 + \bar{t}_{0h} + \hat{t}_{0h}\}$, for all $\gamma = 1, \ldots, \Gamma^t$
    else
      Set $u = (h)$ and $k \leftarrow 1$
      $B_{h\gamma} \leftarrow e_h$, for all $\gamma = 0, \ldots, \Gamma^t$
    $feasible\_path \leftarrow true$
    $j \leftarrow h$
    while $j \neq 2n + 1$ **do**
      $i \leftarrow j$
      $j \leftarrow \arg\max_{j \in \mathcal{N} \setminus \{u\}} \{x_{ij}\}$
      Append j to u
      $k \leftarrow k + 1$
      if $\sum_{i=1}^{k-1} \sum_{j \in \mathcal{L}_i(u)} x_{u_i, u_j} \leq k - 2$ **then**
        Break
      $B_{j\gamma}$ calculated as in (3.2.4)
      if $B_{j\Gamma^t} > l_j$ **then**
        $feasible\_path \leftarrow false$
        $\lambda \leftarrow B_{j\Gamma^t} - l_j$
        for $r \leftarrow k - 2$ **to** 1 **do**
          $\theta \leftarrow B_{u_r, \Gamma^t} - e_{u_r}$
          if $\theta < \lambda$ **then**
            Add RIPEC$(u_r, \ldots, u_k)$ to the pool of inequalities
            $\mathcal{E} \leftarrow \mathcal{E} \cup \{u_1, \ldots, u_r\}$
            Break
        Break
    if $feasible\_path$ **then**
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{u_1, \ldots, u_k\}$
  return pool of inequalities

paths $(i, j, n+i, n+j), (i, j, n+j, n+i), (j, i, n+i, n+j), (j, i, n+j, n+i), (i, n+i, j, n+j), (j, n+j, i, n+i)$ are infeasible.

4.4.2 Pool of inequalities

We incorporate an initial pool of constraints by enumerating valid inequalities with a small set of nodes and all methods are initialised with these constrains. Constraints (4.4.1) to (4.4.10) are based on the subtour elimination constraints. Constraints (4.4.11) to (4.4.14) are based on the precedence between pickup and delivery nodes of a request. Constraints (4.4.15) are generalised order constraints.

$$x_{ij} + x_{ji} \leq 1, \qquad (i,j) \in \mathcal{A} : (j,i) \in \mathcal{A} \quad (4.4.1)$$

$$x_{ij} + x_{ji} + x_{n+i,j} + x_{n+j,i} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.2)$$

$$x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.3)$$

$$x_{ij} + x_{ji} + x_{i,n+i} + x_{j,n+i} + x_{n+i,j} + x_{n+j,i} + x_{n+j,n+i} \leq 2, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.4)$$

$$x_{n+i,n+j} + x_{n+j,n+i} + x_{n+i,j} + x_{n+j,i} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.5)$$

$$x_{i,n+j} + x_{n+j,i} + x_{ij} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.6)$$

$$x_{i,n+j} + x_{n+j,i} + x_{i,n+i} + x_{n+j,n+i} + x_{n+i,n+j} + x_{ij} + x_{n+i,j} \leq 2, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.7)$$

$$x_{n+i,j} + 2x_{j,n+i} + x_{ji} + x_{i,n+i} + x_{n+j,n+i} \leq 2, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.8)$$

$$x_{i,n+i} + x_{n+i,n+j} + x_{n+j,i} + 2x_{i,n+j} + x_{ij} \leq 2, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.9)$$

$$x_{ij} + x_{i,n+i} + x_{i,n+j} + x_{ji} + x_{j,n+i} + x_{j,n+j} + x_{n+i,i} + x_{n+i,j} + x_{n+i,n+j} + x_{n+j,i} + x_{n+j,j} + x_{n+j,n+i} \leq 3, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.10)$$

$$x_{0i} + x_{i,n+j} + x_{n+j,i} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.11)$$

$$x_{i,n+j} + x_{n+j,i} + x_{n+j,2n+1} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.12)$$

$$x_{0i} + x_{i,n+i} + x_{i,n+j} + x_{n+j,i} + x_{n+i,n+j} + x_{n+j,n+i} \leq 2, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.13)$$

$$x_{ij} + x_{ji} + x_{i,n+j} + x_{n+j,i} + x_{j,n+j} + x_{n+j,2n+1} \leq 2, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\} \quad (4.4.14)$$

$$x_{i,n+j} + x_{n+j,i} + x_{n+i,j} + x_{j,n+i} \leq 1, \qquad i \in \mathcal{P}, j \in \mathcal{P} \setminus \{i\}. \quad (4.4.15)$$

Compatibility constraints can be included if a pair of requests are considered incompatible. For $i, j \in \mathcal{P}$, if all paths $(i, j, n+i, n+j), (i, j, n+j, n+i), (j, i, n+i, n+j), (j, i, n+j, n+i), (i, n+i, j, n+j), (j, n+j, i, n+i)$ are infeasible, the following constraints are valid.

$$x_{il} + x_{li} + x_{lj} + x_{jl} \leq 1, \qquad l \in \mathcal{C} \quad (4.4.16)$$

$$x_{il} + x_{li} + x_{l,n+j} + x_{n+j,l} \leq 1, \qquad l \in \mathcal{C} \quad (4.4.17)$$

$$x_{n+i,l} + x_{l,n+i} + x_{lj} + x_{jl} \leq 1, \qquad l \in \mathcal{C} \quad (4.4.18)$$

$$x_{n+i,l} + x_{l,n+i} + x_{l,n+j} + x_{n+j,l} \leq 1, \qquad l \in \mathcal{C}. \quad (4.4.19)$$

We implemented it with additional steps to verify if these paths are robust feasible in the presence of uncertainty.

4.4.3 Initial solution

We provide an initial robust feasible solution to initialise the methods. This initial solution is generated using a simple constructive heuristic based on the worst-case values for all uncertain parameters. The heuristic constructs routes by considering one request at a time, sequentially adding the pickup and delivery nodes of each request in a way that minimally increases the total routing cost. By adding the pickup and delivery nodes sequentially, the vehicle capacity is never exceeded. However, if the time windows of either the pickup or delivery nodes are violated, a new route is started with these nodes. Algorithm 3 depicts this procedure and Table 4 details the data structures used.

Notation	**Description**
$X_{ij} \in \{0, 1\}$	Matrix to initialise the solution
$visited_i \in \{0, 1\}$	Indicates if the pickup node i was already included
$r \in \mathcal{N}$	The last node included
$arrival\_time \geq 0$	The arrival time at the node

Table 4 – Description of the heuristic algorithm's notation for an initial solution.

The procedure first set $r = 0$ as the depot node and initialises the data structures in Lines 2-5. The main loop in Lines 6-18 iterates until all requests are routed. At each iteration, we find the request i with minimum cost $(c_{ri} + c_{i,i+n})$ if included in the route after r. In Line 8, the arrival time is updated considering the worst-case travel time $\bar{t}_{ri} + \hat{t}_{ri}$ to arrive at the pickup node i. If the arrival times at the pickup and delivery nodes do not violate their time windows, Line 9, they are included in the solution (Lines 10 and 11), and the arrival time at the delivery node is updated (Line 12). On the other hand, if the time windows is violated (Lines 16-18), the structures are updated to indicate that the vehicle returned to the depot.

Algorithm 3: Constructive heuristic for initial solution

function `InitialSolution(`*instance_data*`)`
  $r \leftarrow 0$
  $arrival\_time \leftarrow 0$
  $visited_i \leftarrow 0$, for all $i \in \mathcal{P}$
  $X_{ij} \leftarrow 0$, for all $(i,j) \in \mathcal{A}$
  while $\sum_{i \in \mathcal{P}} visited_i < n$ **do**
    $i \leftarrow \arg \min_{i \in \mathcal{P}: visited_i = 0} \{c_{ri} + c_{i,i+n}\}$
    $arrival\_time \leftarrow \max\{e_i, arrival\_time + d_r + \bar{t}_{ri} + \hat{t}_{ri}\}$
    if $arrival\_time \leq l_i$ **and** $arrival\_time + d_i + \bar{t}_{i,i+n} + \hat{t}_{i,i+n} \leq l_{i+n}$ **then**
      $X_{ri} \leftarrow 1$
      $X_{i,i+n} \leftarrow 1$
      $arrival\_time \leftarrow \max\{e_{i+n}, arrival\_time + d_i + \bar{t}_{i,i+n} + \hat{t}_{i,i+n}\}$
      $visited_i = 1$
      $r \leftarrow i + n$
    else
      $X_{r,2n+1} \leftarrow 1$
      $arrival\_time \leftarrow e_0$
      $r \leftarrow 0$
  $X_{r+n,2n+1} = 1$
  return X

Chapter 5
Computational experiments

We assess the performance of the proposed approaches and the robustness of the solutions through extensive computational experiments. The performance of the methods is analysed based on the quality of the solutions in terms of cost and the efficiency at the end of the solution process. The robustness of the solutions is evaluated by examining the trade-off between the additional cost incurred to ensure robustness and the probability of the solution becoming infeasible. The description of the instances and the results are presented in the subsections below. The data is available in the supplementary material at <https://github.com/abreualexp/master-results> .

All experiments were performed on a Linux PC with an Intel Core i7-12700 2.10 GHz processor and 16 GB of RAM, and using the general-purpose optimisation solver Gurobi version 11.0 with its default parameters. The formulations and algorithms were implemented in C++. The time limit for each execution was set to 1 hour (3600 seconds).

5.1 Instances description

In our experiments, we consider the benchmark instances described in Furtado, Munari and Morabito (2017) for the deterministic PDPTW. These instances are grouped into four classes named as follows, based on combinations of vehicle capacities (15 and 20) and time window widths (60 and 120): AA (capacity 15 and time windows width 60); BB (capacity 20 and time windows width 60); CC (capacity 15 and time windows width 120); and DD (capacity 20 and time windows width 120). Each class consists of 15 instances, with the number of requests varying from 5 to 75. The depot is located at the centre of a $[0, 50] \times [0, 50]$ square, and all other nodes are randomly distributed within the same square. A fixed cost of 10^4 is added for each outgoing arc from the depot to enforce the objective of minimising the number of vehicles used, as is common in the literature.

We use different combinations of demand and travel time budgets of uncertainty Γ^q and Γ^t, namely $\{0, 1, 5, 10\}$, and combinations of demand and travel time deviations $\sigma^q, \sigma^t \in \{0, 0.1, 0.2, 0.3\}$, respectively. The deviation values are used to estimate the parameters interval. In total, this leads to 1,644 instances, considering the following uncertainty settings:

(i) $\Gamma^q = \Gamma^t = 0$ and $\sigma^q = \sigma^t = 0$ (deterministic case);

(ii) $\Gamma^q = 1, 5, 10$ and $\Gamma^t = 0$ for each $\sigma^q = 0.1, 0.2, 0.3$ and $\sigma^t = 0$ (demand uncertainty);

(iii) $\Gamma^q = 0$ and $\Gamma^t = 1, 5, 10$ for each $\sigma^q = 0$ and $\sigma^t = 0.1, 0.2, 0.3$ (travel time uncertainty); and

(iv) $\Gamma^q = \Gamma^t = 1, 5, 10$ for each $\sigma^q = \sigma^t = 0.1, 0.2, 0.3$ (both uncertainties).

For each deterministic instance ($\Gamma^q = \Gamma^t = 0$) – a total of 60 – we generate nine additional instances corresponding to the combinations of three levels of the budget of uncertainty and three levels of deviation. However, instances with only 5 requests result in six additional instances, considering budgets of 1 and 5 only.

In some instances considered in our experiments, the worst-case value of individual demands exceeds the vehicle capacity when demand deviations are incorporated, making the instances infeasible. In this scenario, we adapt the instances as follows. For each instance, we check whether the worst-case value of any request demand exceeds the vehicle capacity ($\bar{o}_i + \hat{o}_i > Q$), considering their worst-case value with a deviation of 30% ($\hat{o}_i = \lceil 0.3 \times \bar{o}_i \rceil$, $i \in \mathcal{R}$). If the vehicle capacity is violated, we adjust the nominal demand value of each request i that results in a violation to $\bar{o}_i \coloneqq Q - \lceil 0.3 \times \bar{o}_i \rceil$. Such a transformation is not necessary for travel times, as the worst-case values of travel times did not produce any time window violations in the benchmark instances. The deviation from the nominal demand is then given by $\hat{o}_i = \lceil \sigma^q \times \bar{o}_i \rceil = \hat{q}_i = -\hat{q}_{i+n}$ for all $i \in \mathcal{R}$, and the deviation from the nominal travel time is given by $\hat{t}_{ij} = \lceil \sigma^t \times \bar{t}_{ij} \rceil$ for all $(i, j) \in \mathcal{A}$. The nominal values $\bar{q}_i$ and $\bar{t}_{ij}$ are those provided in the instance data.

5.2 Computational performance

This section presents the performance evaluation of the two proposed approaches, namely the compact formulation within the general-purpose MIP solver of Gurobi (hereafter called `RPDPTW-1`) and the B&C algorithm based on the cutting-plane formulation (hereafter called `RPDPTW-2`). The comparison of their results is conducted in terms of scalability, instance characteristics, and uncertainty levels. Both methods were applied to all the previously described instances, and the main results are summarised in the following tables.

Uncertainty		`RPDPTW-1`			`RPDPTW-2`		
Demand	Travel time	Opt. (%)	Time lim. (%)	Mem. lim. (%)	Opt. (%)	Time lim. (%)	Mem. lim. (%)
		48.3	51.7	0.0	53.3	46.7	0.0
✓		42.4	9.8	47.7	57.8	42.2	0.0
	✓	46.0	54.0	0.0	48.1	51.7	0.2
✓	✓	48.9	8.3	42.8	58.3	41.1	0.6
Overall average		45.9	25.1	29.1	54.7	45.1	0.2

Table 5 – Summary of instance solution status for different uncertainty configuration.

Table 5 presents a summary of the computational performance of each approach when addressing different sources of uncertainty (the instances are grouped by uncertainty configuration). The three chosen performance criteria indicate the percentage of instances that have been solved to optimality, have stopped due to the time limit, or have run out of memory. More precisely, the table reports the percentages of instances for which proven optimal solutions were found within 3600 seconds (*Opt.*), the time limit was reached (*Time lim.*), and the memory limit was reached (*Mem. lim.*). The first row presents the results for the deterministic instances, while the subsequent three rows correspond to uncertainty in demands, travel times, and both. The final row provides the average across all instances. The high proportion of instances that reached memory limit indicates that, in general, `RPDPTW-2` requires less memory to compute compared to `RPDPTW-1`. In particular, this behaviour is more evident in instances with uncertain demands, as they require a large number of additional variables to ensure the pairing property in the formulation. This requirement increases with the budget of uncertainty (see Table 8 for detailed results in terms of Γ^q). The `RPDPTW-2` reached the memory limit in only four instances (one with travel time uncertainty and three with both uncertainties), whereas `RPDPTW-1` reached the memory limit in 478 instances. While `RPDPTW-1` found 20 optimal solutions in cases where `RPDPTW-2` reached the time or memory limit, `RPDPTW-2` found 165 optimal solutions where `RPDPTW-1` reached the time or memory limit.

In Table 6, we analyse the scalability of both formulations across instances of varying sizes. The table presents the average results for each approach, grouped by the number of requests. Specifically, the table shows the proportion of optimal solutions (*Opt.*); the average elapsed computation time to solve instances to optimality (*Time*); the average optimality gap for instances that reached the time or memory limit (*Gap*), as provided by the solver; and the average proportion of the objective value of the solution obtained with the approach relative to the best objective value obtained by any approach (*Ratio*). For each instance and method, the ratio is calculated as: $Ratio = 100 \times \left(\frac{Obj - Obj_{min}}{Obj_{min}}\right)$, where Obj represents the objective value obtained by the approach and $Obj_{min} = \min\{Obj_{RPDPTW-1}, Obj_{RPDPTW-2}\}$ is the best among both methods. It indicates the ability of the approach to find high-quality robust solutions.

The results of computation time represent the average across instances where both approaches found the optimal solution, while the optimality gap results are reported for instances where the approaches reached the time or memory limit. With both approaches, proven optimal solutions were found for all instances with up to 20 requests, with average computation times of 8.6 and 0.7 seconds for `RPDPTW-1` and `RPDPTW-2`, respectively. The `RPDPTW-2` found more optimal solutions across all groups of requests, except for instances with 35 requests. Nevertheless, its average time and ratio remain significantly smaller than those of the `RPDPTW-1`. The largest instance size that `RPDPTW-1` was able to solve extends to 70 requests, where it found an optimal solution for a single instance (class AA, both

n	RPDPTW-1				RPDPTW-2			
	Opt. (%)	Time (s)	Gap (%)	Ratio (%)	Opt. (%)	Time (s)	Gap (%)	Ratio (%)
5	100.0	< 0.1		0.0	100.0	< 0.1		0.0
10	100.0	0.3		0.0	100.0	0.1		0.0
15	100.0	3.9		0.0	100.0	0.6		0.0
20	100.0	27.4		0.0	100.0	1.8		0.0
25	90.2	253.5	0.1	0.4	97.3	76.8	31.9	< 0.1
30	69.6	271.4	23.1	13.0	87.5	141.0	25.1	0.4
35	64.3	126.4	56.3	23.2	61.6	79.6	39.4	< 0.1
40	38.4	119.2	57.5	54.0	49.1	91.1	42.3	1.4
45	15.2	759.9	60.2	79.3	33.9	593.6	51.7	0.8
50	6.3	99.0	60.2	83.3	30.4	37.9	51.7	0.7
55	7.1	185.9	72.0	100.9	24.1	18.6	55.9	0.6
60	6.3	806.4	72.4	116.4	21.4	242.4	52.8	1.6
65	7.1	718.1	75.2	134.6	18.8	272.6	52.8	1.4
70	0.9		73.6	142.3	3.6		50.7	1.6
75	0.0		73.9	153.8	7.1		52.6	0.7

Table 6 – Average results of each method among the different instance sizes.

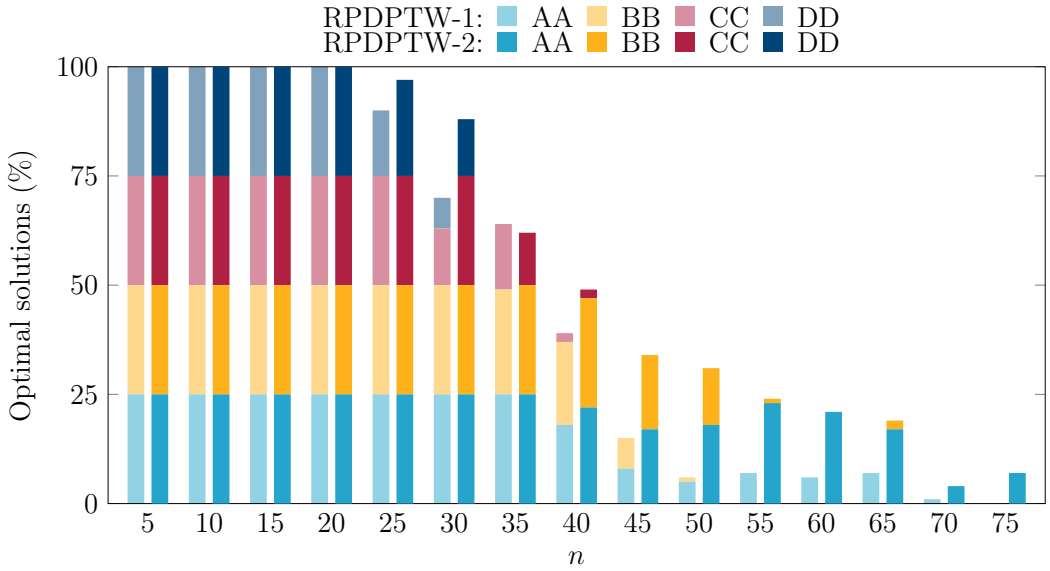
uncertainties, budget of 1, and deviation of 30%). `RPDPTW-2` successfully found optimal solutions for four instances with 70 requests and eight instances with 75 requests. Across the instances where both methods reached optimality, `RPDPTW-1` required an average computation time of 115 seconds, while `RPDPTW-2` took only 53 seconds, representing a reduction of about 50%. As the number of requests increases, the average optimality gap of `RPDPTW-1` increases at a significantly higher rate compared with `RPDPTW-2`. In general, the results of the ratio metric indicate that increasing the instance size does not affect the capability of `RPDPTW-2` to find good feasible solutions as significantly as it affects `RPDPTW-1`.

In Table 7, we analyse how each formulation performs under different uncertainty settings across each instance class (AA, BB, CC, and DD). The table presents the same metrics as those shown in Table 6. `RPDPTW-2` consistently finds more optimal solutions and requires less computation time than `RPDPTW-1`. In instances of the AA class, `RPDPTW-2` finds more optimal solutions for all uncertainty configurations and requires less computation time. Only in instances with travel time uncertainty does `RPDPTW-2` present a larger optimality gap and ratio. A similar behaviour is observed in classes BB and CC. However, in these instances, `RPDPTW-2` also finds fewer optimal solutions in the case of travel time uncertainty. In the DD class, `RPDPTW-2` does not perform as well in terms of optimal solutions only for instances with no uncertainty in the DD class. Nevertheless, even under travel time uncertainty, `RPDPTW-2` solves more instances and presents a better ratio, although it requires more computation time. For each uncertainty setting, we observe that the proportion of optimal solutions for both methods decreases gradually as instance classes progress from AA to DD. Notably, increasing the time window length has a more significant impact on the proportion of optimal solutions found compared with increasing capacity.

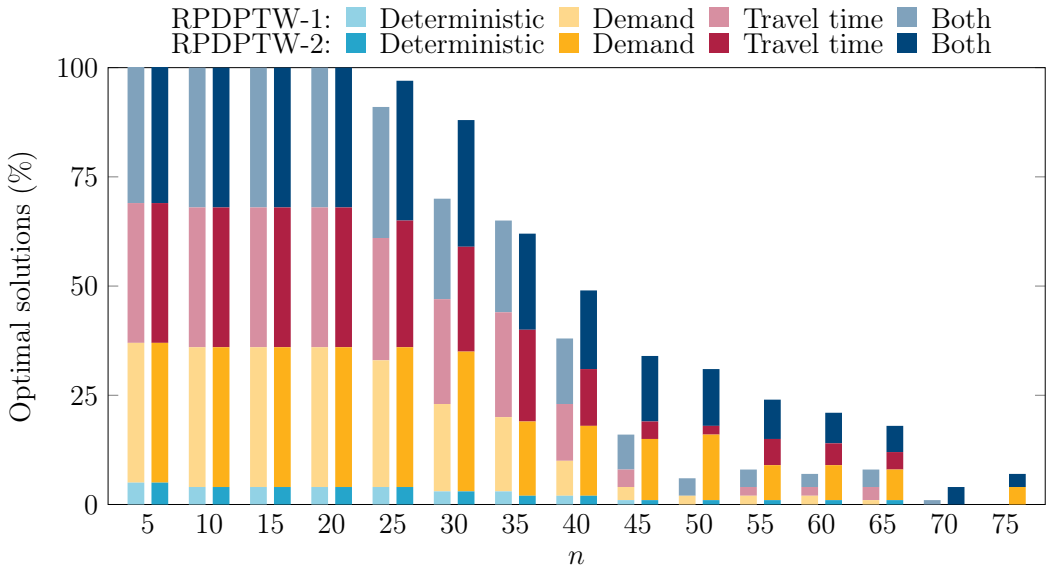
Class	Uncertainty		RPDPTW-1				RPDPTW-2			
	Demand	Travel time	Opt. (%)	Time (s)	Gap (%)	Ratio (%)	Opt. (%)	Time (s)	Gap (%)	Ratio (%)
AA			53.3	45.8	20.8	2.1	80.0	2.8	12.8	< 0.1
	✓		55.3	42.6	59.5	63.5	88.6	4.6	18.0	0.0
		✓	56.8	156.8	7.2	< 0.1	67.4	60.5	15.1	1.2
	✓	✓	68.2	14.3	65.9	53.1	86.4	4.7	13.3	< 0.1
BB			60.0	371.3	21.7	< 0.1	60.0	320.0	24.5	< 0.1
	✓		49.2	81.7	63.8	75.3	64.4	7.0	14.4	< 0.1
		✓	53.8	34.0	11.4	0.2	53.0	116.1	22.6	1.0
	✓	✓	53.8	46.4	57.6	71.9	65.9	14.0	16.4	0.1
CC			40.0	1.6	88.7	< 0.1	40.0	1.6	73.6	1.3
	✓		33.3	98.4	89.6	103.5	39.4	1.9	65.6	< 0.1
		✓	45.5	151.9	77.8	1.4	42.4	168.3	73.7	2.2
	✓	✓	40.9	243.9	88.2	79.8	45.5	45.1	71.9	0.7
DD			40.0	182.6	71.8	1.6	33.3	48.9	69.3	< 0.1
	✓		31.8	272.8	88.9	164.1	38.6	25.8	59.4	< 0.1
		✓	28.0	133.6	66.3	2.9	29.5	179.3	66.3	2.6
	✓	✓	32.6	278.4	85.7	148.7	35.6	40.0	61.3	< 0.1

Table 7 – Average results for instance classes and uncertainty configurations.

`RPDPTW-2` was faster than `RPDPTW-1` in over 88% of the instances where both methods found optimal solutions. Among the 85 instances where `RPDPTW-1` was faster, 63 involved only travel time uncertainty. Despite the overall average computation time of `RPDPTW-2` being significantly lower than that of `RPDPTW-1`, this trend does not hold for instances with travel time uncertainty. On average, the computation time for `RPDPTW-1` on instances with uncertain travel times is about 116 seconds, while for `RPDPTW-2`, it is around 121 seconds. In the case of demand uncertainty, the average computation times for `RPDPTW-1` and `RPDPTW-2` are approximately 108 and 9 seconds, respectively. When considering both uncertainties, the average computation times are 115 seconds for `RPDPTW-1` and 22 seconds for `RPDPTW-2`.



(a) Optimal solutions for each instance class.



(b) Optimal solutions for each uncertainty configuration.

Figure 2 – Optimal solutions (in percentage) with each method for different instance sizes and characteristics.

Figures 2a and 2b illustrate the scalability of optimal solutions found, measured by the number of requests. The light-coloured bars (on the left) represent results of `RPDPTW-1`, while the dark-coloured bars (on the right) represent results of `RPDPTW-2`. In Figure 2a, results are grouped into classes, with the stacked bars (from bottom to top) corresponding to classes AA, BB, CC, and DD. In Figure 2b, results are grouped by uncertainty configuration, where the stacked bars (from bottom to top) represent deterministic instances (no uncertainty), uncertain demands, uncertain travel times, and both uncertainties. Both `RPDPTW-1` and `RPDPTW-2` are capable of finding optimal solutions for instances in classes DD and CC with up to 30 and 40 requests, respectively. We can observe that longer time window lengths have the most significant impact on the methods' performance. Notably, the instances with 75 requests solved to optimality belong to class AA and involve uncertainty in either demands or both parameters. As the number of requests increases, the proportion of instances with travel time uncertainty optimally solved by `RPDPTW-2` decreases more rapidly than those with demand uncertainty or both uncertainties. From Figure 2b, we observe that `RPDPTW-1` generates optimal solutions for deterministic instances only up to 45 requests, whereas `RPDPTW-2` successfully solves instances with up to 65 requests.

In Table 8, we analyse how both formulations behave under different uncertainty levels. The first two columns present the various combinations of uncertainty budgets. In the first row, results for deterministic instances ($\Gamma^q = \Gamma^t = 0$) are shown. The subsequent rows display results for the three levels of the uncertainty budget (1, 5, and 10). Rows two to four correspond to the demand uncertainty configuration, while rows five to seven refer to travel time uncertainty. Results for instances with both uncertainties are presented in the final three rows. `RPDPTW-2` solved more instances to optimality across all uncertainty levels. As previously discussed, in the travel time uncertainty scenario, `RPDPTW-2` required more computation time compared to `RPDPTW-1`, achieving a higher ratio when $\Gamma^t = 1, 5$. In general, as the level of uncertainty increases, solving the instances becomes more challenging. The uncertainty budget directly affects the sizes of variables and constraints in `RPDPTW-1`, as well as the size of data structures in the separation procedures of `RPDPTW-2`. This trend is reflected in the metrics *Opt.*, *Gap.*, and *Ratio*. However, an exception occurs in the case of travel time uncertainty, where performance tends to improve as the uncertainty budget increases from 1 to 5.

5.3 Robustness analysis

In this section, we evaluate the robustness of solutions using two metrics: *the Price of Robustness* (*PoR*) and *Risk*. The *PoR* represents the percentage of additional routing cost incurred to ensure feasibility under uncertainty compared to the cost of the deterministic solution, and it is calculated as: $PoR = 100 \times \left(\frac{Obj_{Robust} - Obj_{Deterministic}}{Obj_{Deterministic}} \right)$, where Obj_{Robust} corresponds to the routing cost associated with the robust solution, whereas

Γ^q	Γ^t	RPDPTW-1				RPDPTW-2			
		Opt. (%)	Time (s)	Gap (%)	Ratio (%)	Opt. (%)	Time (s)	Gap (%)	Ratio (%)
0	0	48.3	171.4	59.5	0.9	53.3	116.9	53.8	0.3
1	0	51.7	139.9	63.5	12.8	57.2	19.8	49.2	< 0.1
5	0	41.1	87.1	89.5	129.3	59.4	1.5	49.5	0.0
10	0	33.9	85.8	93.0	167.1	56.5	0.9	49.4	0.0
0	1	46.1	97.3	45.4	0.7	48.9	122.0	47.9	1.4
0	5	48.3	113.9	45.5	0.5	51.1	105.2	51.5	2.3
0	10	43.5	139.5	51.7	2.2	44.0	138.5	51.4	1.4
1	1	56.7	134.4	63.4	8.8	58.3	48.0	52.5	0.6
5	5	49.4	125.1	87.0	114.0	61.1	6.8	52.5	< 0.1
10	10	39.9	71.9	86.8	146.1	55.4	2.6	51.3	< 0.1

Table 8 – Average results of each method for the different levels of uncertainty.

$Obj_{Deterministic}$ corresponds to the routing cost of the deterministic solution. The *Risk* refers to the experimental probability of a solution becoming infeasible and is estimated through Monte Carlo simulations. This analysis provides valuable insights to assist decision-makers in defining parameters while considering the different problem structures they might encounter. In our analyses, we considered instances with 10, 15, and 20 requests (336 instances in total), as this range represents the threshold where optimal solutions were consistently found across all instances. Instances with 5 requests were excluded because experiments were not conducted with a budget of uncertainty equal to 10.

The Monte Carlo simulation approach is designed to evaluate the risk of a solution against the possible data deviations. The procedure involves assessing the frequency with which a solution becomes infeasible, regardless of the restrictions imposed by the uncertainty set. We performed 10,000 simulations for each solution obtained across the 336 instances considered. In each simulation, values for the demands of the requests and arc travel times are generated, assuming that the realization of uncertain parameters follows a uniform distribution within their respective intervals. For the instances associated with an uncertainty configuration (324 instances) –whether related to demand ($\Gamma^q > 0$), travel time ($\Gamma^t > 0$), or both ($\Gamma^q > 0$ and $\Gamma^t > 0$)– we consider the deviation levels used when solving the instances, specifically $\sigma^q, \sigma^t \in \{0.1, 0.2, 0.3\}$. Hence, we run the simulations for each instance solution $x^{\sigma,\mathbf{\Gamma}}$, where σ represents the pair of deviations (σ^q, σ^t) associated with the instance, and $\mathbf{\Gamma}$ represents the pair of uncertainty budgets (Γ^q, Γ^t) associated with the instance. For each specific instance, the simulation procedure can be summarized in Algorithm 4. In the case of deterministic instances (12 instances), where no deviation levels or uncertainty budgets are considered, we adapted the simulation procedure. Specifically, Algorithm 4 was executed nine times, with each run corresponding to one of the combinations of uncertainty type (demand, travel time, or both) and deviation level (10%, 20%, or 30%): $(\Gamma^q > 0, \Gamma^t > 0, \Gamma^q > 0$ and $\Gamma^t > 0) \times (0.1, 0.2, 0.3)$.

Table 9 presents the average results of *PoR* and *Risk* for each combination of

Algorithm 4: Risk Computation

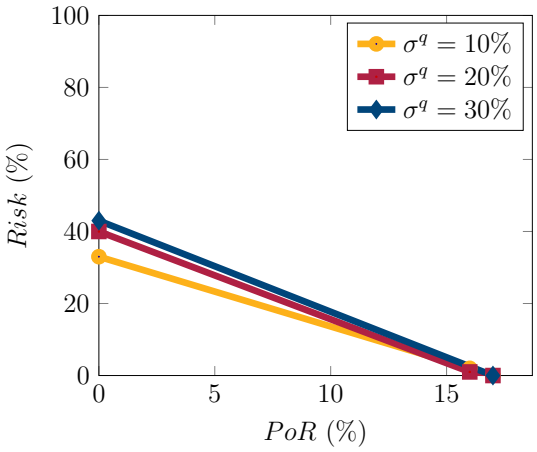
```
1  function RiskComputation(x^{σ,Γ})
2      violation_count ← 0
3      for k ← 1 to 10,000 do
4          õ ← ō
5          t̃ ← t̄
6          if Γ^q > 0 then
7              foreach i ∈ R do
8                  õ_i ← U[ō_i − ⌈σ^q × ō_i⌉, ō_i + ⌈σ^q × ō_i⌉]
9          if Γ^t > 0 then
10             foreach (i, j) ∈ A do
11                 t̃_ij ← U[t̄_ij − ⌈σ^t × t̄_ij⌉, t̄_ij + ⌈σ^t × t̄_ij⌉]
12         if x^{σ,Γ} is infeasible for (õ, t̃) then
13             violation_count ← violation_count + 1
14     risk ← violation_count / 10,000
15     return risk
```

uncertainty budget and deviation level. Moreover, the last two columns display the average results across all deviation values. A *PoR* or *Risk* value of zero is represented by a "-" in the table. The first set of experiments (first four rows) considers uncertainty in demands, the second set (rows five to seven) focuses on travel time uncertainty, and the third set (last four rows) accounts for both uncertainties. As expected, the value of *Risk* increases as the deviation level rises regarding deterministic solutions. In general, deterministic solutions exhibit an overall average *Risk* of about 32%. The average *Risk* when considering instances with uncertain travel times is smaller compared to other uncertainty configurations. The *PoR* of robust solutions also increases with higher deviation levels, as the worst-case parameter values escalate and impact the capacity or time windows. However, we observe no consistent *Risk* pattern in robust solutions with the increase of deviation levels. In other words, we cannot say that the risk always increases or decreases as the deviation varies. This unpredictability arises because robust solutions for higher deviations may exhibit differing levels of flexibility in accommodating additional deviations. Yet, the *Risk* is significantly reduced when robust solutions are obtained with larger uncertainty budgets. In fact, our experiments indicate an overall average probability of infeasibility of less than 3% for robust solutions considering a budget of 1, with an average PoR of 7%. This probability drops to zero when the uncertainty budget increases to 5 while the solutions incurred an average PoR of 13%. It is worth noting that the additional cost of ensuring robustness for up to five worst-case parameter values, as opposed to up to one, grows at a higher rate in instances with uncertain travel times compared to other uncertainty settings.

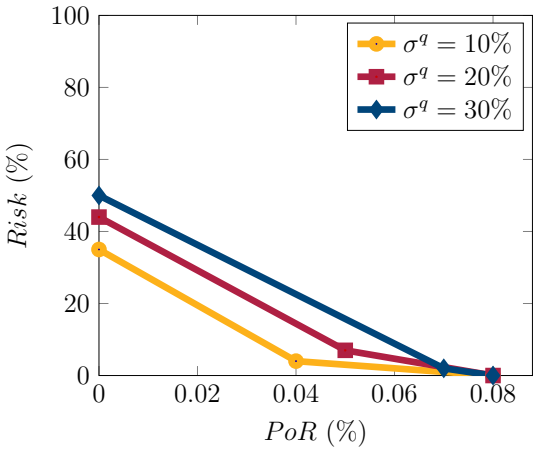
Γ^q	Γ^t	10% deviation		20% deviation		30% deviation		Overall	
		PoR (%)	Risk (%)	PoR (%)	Risk (%)	PoR (%)	Risk (%)	PoR (%)	Risk (%)
0	0	-	33.98	-	41.77	-	46.28	-	40.68
1	0	8.10	3.01	8.11	4.19	8.12	1.04	8.11	2.75
5	0	8.12	-	8.12	-	8.12	-	8.12	-
10	0	8.12	-	8.12	-	8.12	-	8.12	-
0	0	-	8.68	-	21.77	-	31.92	-	20.79
0	1	0.02	1.49	0.03	1.70	0.04	3.72	0.03	2.30
0	5	0.03	-	4.06	-	24.14	-	9.41	-
0	10	0.04	-	4.07	-	24.15	-	9.42	-
0	0	-	40.96	-	56.85	-	65.59	-	54.47
1	1	8.12	6.26	12.12	5.24	12.15	4.48	10.80	5.33
5	5	12.14	-	16.17	-	24.22	-	17.51	-
10	10	12.14	-	16.17	-	24.22	-	17.51	-

Table 9 – PoR and Risk (in percentage) for different deviation and uncertainty levels.

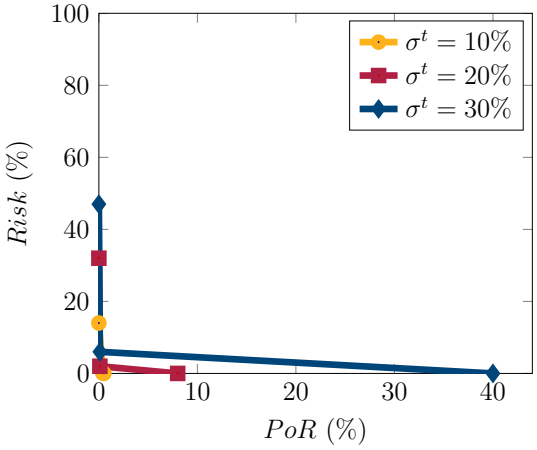
Figure 3 shows the relation between *PoR* and *Risk* grouped in instances with varying values of capacity and time windows length. We can now observe how different capacity and time windows length influence the cost and the feasibility of robust solutions. In particular, we show the results for instances with: Tight capacity (classes AA and CC) and uncertain demands (Figure 3a); Tight time windows (classes AA and BB) and uncertainty travel times (Figure 3c); Tight capacity and time windows (class AA) and both uncertainties (Figure 3e); Loose capacity (classes BB and DD) and uncertain demands (Figure 3b); Loose time windows (classes CC and DD) and uncertain travel times (Figure 3d); and, finally, loose capacity and time windows (class DD) and both uncertainties (Figure 3f). The series in each plot represent the different deviation levels of the considered uncertainty. Specifically, the deviations are represented as follows: 10% (yellow lines with circle markers), 20% (red lines with square markers), and 30% (blue lines with diamond markers). In general, we observe that the *PoR* incurred by robust solutions in instances with either tight time windows or tight capacity is significantly higher when compared to instances with either loose time windows or loose capacity. This suggests that, in more restrictive instances, routes may require substantial changes to ensure robustness, whereas in less restrictive instances, only minor adjustments to the solution are needed. The *Risk* of deterministic solutions, with respect to uncertain demands, shows no significant difference between instances with tight or loose capacity. This indicates that the proportion of vehicle occupancy remains approximately the same, regardless of capacity constraints. On the other hand, less restrictive instances are associated with lower *Risk* across other uncertainty settings. This suggests that looser instances allow for more flexible solutions, which are better equipped to handle variations in demand and travel time uncertainty.



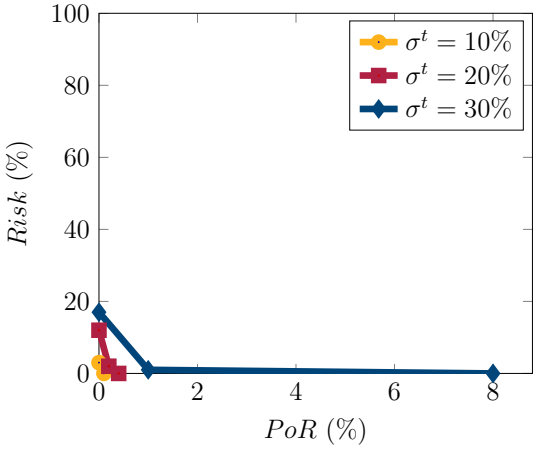
(a) Tight capacity under demand uncertainty



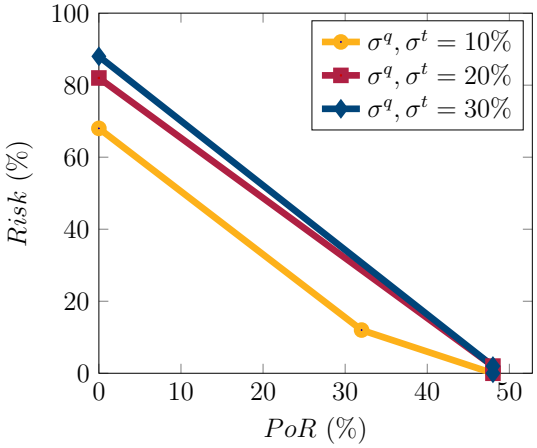
(b) Loose capacity under demand uncertainty



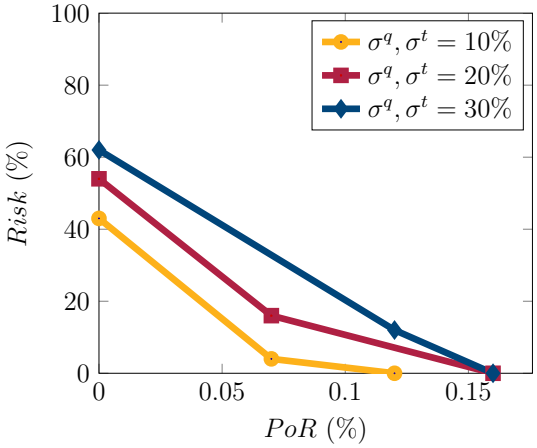
(c) Tight time windows under travel time uncertainty



(d) Loose time windows under travel time uncertainty



(e) Tight capacity and time windows under both uncertainties



(f) Loose capacity and time windows under both uncertainties

Figure 3 – Trade-off between PoR and Risk for different types of instances and uncertainty configuration

Chapter 6
Conclusions

Our study introduced the robust pickup and delivery problem with time windows (RPDPTW), where uncertainty in demands and travel times was incorporated into the deterministic counterpart using robust optimisation (RO). Demands and travel times realizations are assumed to belong to cardinality-constrained uncertainty sets. This problem extends the deterministic PDPTW by ensuring solution feasibility under uncertain conditions while balancing cost and conservatism. Recursive equations were introduced to verify whether a route solution remains robust feasible under demand uncertainty.

The literature review shows that although the pickup and delivery problem is a prominent topic in operations research, it has been barely researched under the robust optimisation perspective, specially in its classical format. Robust pickup and delivery problems were mainly studied in one-to-many-to-one and many-to-many formats and in rich variants. To our knowledge, one publication so far attempted to solve one-to-one PDPTW variant, which considers electric vehicles, under uncertainty. The paper incorporated demand uncertainty and modelled it in discrete scenarios. We are not aware of any other paper that addresses demand and travel time uncertainty jointly in the PDPTW via RO and the cardinality-constrained uncertainty set.

To address the RPDPTW, we proposed two robust counterpart formulations: one compact and one based on cutting-planes. The compact robust counterpart formulation was derived by incorporating the linearisation of the recursive equations, enabling straightforward implementation with general-purpose MIP solvers. The cutting-plane robust counterpart formulation includes an exponential number of constraints to guarantee robust feasibility. A tailored B&C algorithm was developed to efficiently handle this formulation, leveraging the problem's specific characteristics and the structure of the recursive equations derived from dynamic programming.

We conducted extensive computational experiments to evaluate the performance of using the proposed formulations to solve the RPDPTW. Deterministic benchmark instances were adapted to accommodate the considered uncertainties and were solved using different levels of robustness. The results obtained using the cutting-plane formulation within the tailored B&C method indicate the superior performance of this approach, as more instances were solved to proven optimality in less computation time while providing high-quality solutions for the remaining instances. In most cases where using the compact formulation led to a better performance, the instances were related to travel time uncertainty. On the other hand, the presence of demand uncertainty in the compact formulation resulted in a large number of binary variables. This is due to the need of pairing pickup and delivery

nodes in the problem.

The robustness of the solutions obtained with the proposed approaches was evaluated through a series of Monte Carlo simulations. In these experiments, we collected metrics for the additional routing cost, called as the Price of Robustness (PoR), and the probability of a solution becoming infeasible ($Risk$). In summary, the results indicate that robust solutions exhibit a significantly lower $Risk$ compared to deterministic solutions. The PoR and $Risk$ can display varying behaviours depending on instance characteristics and uncertainty settings. In particular, we analysed how more or less restricted instances, with respect to capacity and time windows, are affected by different levels of uncertainty.

References

Agra, A. et al. Layered formulation for the robust vehicle routing problem with time windows. In: Mahjoub, A. et al. (Ed.). *Combinatorial Optimization*. [S.l.]: Springer, 2012. v. 7422, p. 249–260.

Agra, A. et al. The robust vehicle routing problem with time windows. *Computers & Operations Research*, Pergamon, v. 40, p. 856–866, Mar. 2013.

Allahyari, S.; Yaghoubi, S.; Woensel, T. V. The secure time-dependent vehicle routing problem with uncertain demands. *Computers & Operations Research*, Pergamon, v. 131, p. 105253, Jul. 2021.

Alyasiry, A. M.; Forbes, M.; Bulmer, M. An exact algorithm for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science*, v. 53, p. 1695–1705, Nov. 2019.

Ascheuer, N.; Fischetti, M.; Grötschel, M. A polyhedral study of the asymmetric traveling salesman problem with time windows. 2000.

Baldacci, R.; Bartolini, E.; Mingozzi, A. An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, INFORMS, v. 59, p. 414–426, Apr. 2011.

Battarra, M.; Cordeau, J.-F.; Iori, M. Chapter 6: Pickup-and-delivery problems for goods transportation. In: Toth, P.; Vigo, D. (Ed.). *Vehicle Routing*. [S.l.]: Society for Industrial and Applied Mathematics, 2014. p. 161–191.

Battarra, M. et al. The traveling salesman problem with pickups, deliveries, and handling costs. *Transportation Science*, INFORMS, v. 44, p. 383–399, Aug. 2010.

Ben-tal, A.; Nemirovski, A. Robust solutions of uncertain linear programs. *Operations Research Letters*, North-Holland, v. 25, p. 1–13, Aug. 1999.

Bent, R.; Van Hentenryck, P. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, Pergamon, v. 33, p. 875–893, Apr. 2006.

Berbeglia, G. et al. Static pickup and delivery problems: a classification scheme and survey. *TOP*, Springer, v. 15, p. 1–31, Jul. 2007.

Berbeglia, G.; Cordeau, J. F.; Laporte, G. Dynamic pickup and delivery problems. *European Journal of Operational Research*, North-Holland, v. 202, p. 8–15, Apr. 2010.

Bertsimas, D.; Sim, M. The price of robustness. *Operations Research*, INFORMS, v. 52, p. 35–53, Feb. 2004.

Cai, J. et al. A survey of dynamic pickup and delivery problems. *Neurocomputing*, Elsevier, v. 554, p. 126631, Oct. 2023.

Campos, R.; Coelho, L. C.; Munari, P. New formulations for the robust vehicle routing problem with time windows under demand and travel time uncertainty. *OR Spectrum*, Springer Science and Business Media Deutschland GmbH, p. 1–43, Jul. 2024.

Ceria, S.; Stubbs, R. A. Incorporating estimation errors into portfolio selection: Robust portfolio construction. *Journal of Asset Management 2006 7:2*, Palgrave, v. 7, p. 109–127, Jul. 2006.

Chami, Z. A. et al. A robust pickup and delivery problem with uncertain travel time. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, IEEE Computer Society, v. 2018-Novem, p. 940–946, Dec. 2018.

Chami, Z. A. et al. A grasp-alns combination for robust pickup and delivery problem. *International Journal of Production Research*, Taylor & Francis, v. 60, p. 3809–3828, Jun. 2022.

Chardy, M.; Klopfenstein, O. Handling uncertainties in vehicle routing problems through data preprocessing. *Transportation Research Part E: Logistics and Transportation Review*, Pergamon, v. 48, p. 667–683, May 2012.

Cherkesly, M. et al. Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. *European Journal of Operational Research*, North-Holland, v. 250, p. 782–793, May 2016.

Christiaens, J.; Berghe, G. V. Slack induction by string removals for vehicle routing problems. *Transportation Science*, INFORMS, v. 54, p. 417–433, Mar. 2020.

Cordeau, J.-F. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, INFORMS, v. 54, p. 573–586, Jun. 2006.

Cordeau, J. F.; Laporte, G. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, Springer, v. 153, p. 29–46, Sep. 2007.

Cordeau, J.-F.; Laporte, G.; Ropke, S. Recent models and algorithms for one-to-one pickup and delivery problems. In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. [S.l.]: Springer US, 2008. v. 43, p. 327–357.

Curtois, T. et al. Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows. *EURO Journal on Transportation and Logistics*, Elsevier, v. 7, p. 151–192, Jun. 2018.

Dahle, L. et al. The pickup and delivery problem with time windows and occasional drivers. *Computers & Operations Research*, Pergamon, v. 109, p. 122–133, Sep. 2019.

De La Vega, J.; Munari, P.; Morabito, R. Robust optimization for the vehicle routing problem with multiple deliverymen. *Central European Journal of Operations Research*, Springer Verlag, v. 27, p. 905–936, Dec. 2019.

De La Vega, J.; Munari, P.; Morabito, R. Exact approaches to the robust vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, Pergamon, v. 124, p. 105062, Dec. 2020.

Desaulniers, G. et al. 9. vrp with pickup and delivery. In: *The Vehicle Routing Problem*. [S.l.]: Society for Industrial and Applied Mathematics, 2002. p. 225–242.

Doerner, K. F.; Salazar-gonzález, J.-J. Chapter 7: Pickup-and-delivery problems for people transportation. In: Toth, P.; Vigo, D. (Ed.). *MOS-SIAM Series on Optimization*. [S.l.]: Society for Industrial and Applied Mathematics, 2014. p. 193–212.

Drexl, M. On the one-to-one pickup-and-delivery problem with time windows and trailers. *Central European Journal of Operations Research*, Springer Science and Business Media Deutschland GmbH, v. 29, p. 1115–1162, Sep. 2021.

Dumas, Y.; Desrosiers, J.; Soumis, F. The pickup and delivery problem with time windows. *European Journal of Operational Research*, v. 54, p. 7–22, Sep. 1991.

Erdoğan, G. et al. Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs. *Computers & Operations Research*, Pergamon, v. 39, p. 1074–1086, May 2012.

Furtado, M. G. S.; Munari, P.; Morabito, R. Pickup and delivery problem with time windows: A new compact two-index formulation. *Operations Research Letters*, North-Holland, v. 45, p. 334–341, Jul. 2017.

Garvin, W. W. et al. Applications of linear programming in the oil industry. *Management Science*, INFORMS, v. 3, p. 407–430, Jul. 1957.

Gasque, D.; Munari, P. Metaheuristic, models and software for the heterogeneous fleet pickup and delivery problem with split loads. *Journal of Computational Science*, Elsevier, v. 59, p. 101549, Mar. 2022.

Gendreau, M.; Jabali, O.; Rei, W. 50th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, INFORMS, v. 50, p. 1163–1173, Nov. 2016.

Ghilas, V. et al. Branch-and-price for the pickup and delivery problem with time windows and scheduled lines. *Transportation Science*, INFORMS, v. 52, p. 1191–1210, Oct. 2018.

Goeke, D. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. *European Journal of Operational Research*, North-Holland, v. 278, p. 821–836, Nov. 2019.

Golsefidi, A. H.; Jokar, M. R. A. A robust optimization approach for the production-inventory-routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, Pergamon, v. 143, p. 106388, May 2020.

Gounaris, C. E.; Wiesemann, W.; Floudas, C. A. The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, INFORMS, v. 61, p. 677–693, Jun. 2013.

Gribkovskaia, I.; Laporte, G. One-to-many-to-one single vehicle pickup and delivery problems. In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. [S.l.]: Springer US, 2008. v. 43, p. 359–377.

Gschwind, T. A comparison of column-generation approaches to the synchronized pickup and delivery problem. *European Journal of Operational Research*, North-Holland, v. 247, p. 60–71, Nov. 2015.

Gschwind, T. et al. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. *European Journal of Operational Research*, North-Holland, v. 266, p. 521–530, Apr. 2018.

Guo, F. et al. Robust optimization of microhub network and mixed service strategy for a multidepot location-routing problem. *Computers & Industrial Engineering*, Pergamon, v. 190, p. 110070, Apr. 2024.

Hansuwa, S.; Kumar, M. R. V.; Chandrasekharan, R. Analysis of box and ellipsoidal robust optimization, and attention model based reinforcement learning for a robust vehicle routing problem. *Sadhana - Academy Proceedings in Engineering Sciences*, Springer, v. 47, p. 1–23, Jun. 2022.

Hernández-pérez, H.; Salazar-gonzález, J.-J. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science*, INFORMS, v. 38, p. 245–255, May 2004.

Hernández-pérez, H.; Salazar-gonzález, J. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks*, John Wiley & Sons, Ltd, v. 50, p. 258–272, Dec. 2007.

Huang, X. et al. Robust optimization model of feeder lines routing based on the hub port. *Transportation Journal*, Duke University Press, v. 59, p. 279–303, Jul. 2020.

Jaillet, P.; Qi, J.; Sim, M. Routing optimization under uncertainty. *Operations Research*, INFORMS, v. 64, p. 186–200, Feb. 2016.

Kalantari, B.; Hill, A. V.; Arora, S. R. An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research*, North-Holland, v. 22, p. 377–386, Dec. 1985.

Kallehauge, B.; Boland, N.; Madsen, O. B. Path inequalities for the vehicle routing problem with time windows. *Networks*, John Wiley & Sons, Ltd, v. 49, p. 273–293, 7 2007. ISSN 0028-3045.

Koç, Ç.; Laporte, G. Vehicle routing with backhauls: Review and research perspectives. *Computers & Operations Research*, Pergamon, v. 91, p. 79–91, Mar. 2018.

Koç, Ç.; Laporte, G.; Tükenmez, İ. A review of vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, Pergamon, v. 122, p. 104987, Oct. 2020.

Lau, H. C.; Liang, Z. Pickup and delivery with time windows: algorithms and test case generation. *International Journal on Artificial Intelligence Tools*, World Scientific Publishing Company, v. 11, p. 455–472, Sep. 2002.

Lee, C.; Lee, K.; Park, S. Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society*, v. 63, p. 1294–1306, Sep. 2012.

Letchford, A. N.; Salazar-gonzález, J. J. Stronger multi-commodity flow formulations of the capacitated vehicle routing problem. *European Journal of Operational Research*, North-Holland, v. 244, p. 730–738, Aug. 2015.

Li, H.; Lim, A. A metaheuristic for the pickup and delivery problem with time windows. In: *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence*. [S.l.]: IEEE Comput. Soc, 2001. p. 160–167.

Liu, X. et al. Robust optimization for the electric vehicle pickup and delivery problem with time windows and uncertain demands. *Computers & Operations Research*, Pergamon, v. 151, p. 106119, Mar. 2023.

Lu, Q.; Dessouky, M. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, INFORMS, v. 38, p. 503–514, Nov. 2004.

Lu, Q.; Dessouky, M. M. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, North-Holland, v. 175, p. 672–687, Dec. 2006.

Lyu, Z.; Yu, A. J. The pickup and delivery problem with transshipments: Critical review of two existing models and a new formulation. *European Journal of Operational Research*, North-Holland, v. 305, p. 260–270, Feb. 2023.

Masson, R. et al. A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. *European Journal of Operational Research*, North-Holland, v. 236, p. 849–862, Aug. 2014.

Meng, S.; Chen, Y.; Li, D. The multi-visit drone-assisted pickup and delivery problem with time windows. *European Journal of Operational Research*, North-Holland, v. 314, p. 685–702, Apr. 2024.

Mitrovic-minic, S. *Pickup and Delivery Problem with Time Windows: A Survey*. 1998.

Mondal, A.; Roy, S. K. Multi-objective sustainable opened- and closed-loop supply chain under mixed uncertainty during covid-19 pandemic situation. *Computers & Industrial Engineering*, Pergamon, v. 159, p. 107453, Sep. 2021.

Mousavi, S. M.; Vahdani, B. A robust approach to multiple vehicle location-routing problems with time windows for optimization of cross-docking under uncertainty. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 32, p. 49–62, Jan. 2017.

Munari, P. et al. The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, INFORMS, v. 53, p. 1043–1066, Jul. 2019.

Naccache, S.; Côté, J. F.; Coelho, L. C. The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, North-Holland, v. 269, p. 353–362, Aug. 2018.

Nagata, Y.; Kobayashi, S. Guided ejection search for the pickup and delivery problem with time windows. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.]: Springer, Berlin, Heidelberg, 2010. v. 6022 LNCS, p. 202–213.

Nanry, W. P.; Barnes, J. W. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, Pergamon, v. 34, p. 107–121, Feb. 2000.

Nowak, M.; Ergun Özlem; White, C. C. Pickup and delivery with split loads. *Transportation Science*, INFORMS, v. 42, p. 32–43, Feb. 2008.

Ordóñez, F. Robust vehicle routing. In: *Risk and Optimization in an Uncertain World*. [S.l.]: INFORMS, 2010. p. 153–178.

Pankratz, G. A grouping genetic algorithm for the pickup and delivery problem with time windows. *OR Spectrum*, Springer, v. 27, p. 21–41, Jan. 2005.

Parragh, S. N.; Doerner, K. F.; Hartl, R. F. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, v. 58, p. 21–51, Apr. 2008.

Parragh, S. N.; Doerner, K. F.; Hartl, R. F. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, v. 58, p. 81–117, Jun. 2008.

Pavlova, E. S.; Shichiyakh, R. A. A robust mathematical model for vehicle routing problem with simultaneous pickup and delivery and worker allocation. *Industrial Engineering & Management Systems*, Korean Institute of Industrial Engineers, v. 20, p. 201–212, Jun. 2021.

Pessoa, A. et al. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, Springer, v. 183, p. 483–523, Sep. 2020.

Pourmohammad-zia, N. et al. A robust optimization approach for platooning of automated ground vehicles in port hinterland corridors. *Computers & Industrial Engineering*, Elsevier Ltd, v. 177, p. 109046, Mar. 2023.

Praxedes, R. et al. A unified exact approach for a broad class of vehicle routing problems with simultaneous pickup and delivery. *Computers & Operations Research*, Pergamon, v. 162, p. 106467, Feb. 2024.

Qu, Y.; Bard, J. F. A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, INFORMS, v. 49, p. 254–270, May 2015.

Ropke, S.; Cordeau, J.-F. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, INFORMS, v. 43, p. 267–286, Aug. 2009.

Ropke, S.; Cordeau, J. F.; Laporte, G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, John Wiley & Sons, Ltd, v. 49, p. 258–272, Jul. 2007.

Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, INFORMS, v. 40, p. 455–472, Nov. 2006.

Santos, M. J. et al. A robust optimization approach for the vehicle routing problem with selective backhauls. *Transportation Research Part E: Logistics and Transportation Review*, Pergamon, v. 136, p. 101888, Apr. 2020.

Sartori, C. S.; Buriol, L. S. A study on the pickup and delivery problem with time windows: Matheuristics and new instances. *Computers & Operations Research*, Pergamon, v. 124, p. 105065, Dec. 2020.

Savelsbergh, M.; Sol, M. Drive: Dynamic routing of independent vehicles. *Operations Research*, INFORMS, v. 46, p. 474–490, Aug. 1998.

Savelsbergh, M. W. P.; Sol, M. The general pickup and delivery problem. *Transportation Science*, INFORMS, v. 29, p. 17–29, Feb. 1995.

Solano-charris, E.; Prins, C.; Santos, A. C. Local search based metaheuristics for the robust vehicle routing problem with discrete scenarios. *Applied Soft Computing*, Elsevier, v. 32, p. 518–531, Jul. 2015.

Soyster, A. L. Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, INFORMS, v. 21, p. 1154–1157, Oct. 1973.

Subramanyam, A.; Repoussis, P. P.; Gounaris, C. E. Robust optimization of a broad class of heterogeneous vehicle routing problems under demand uncertainty. *INFORMS Journal on Computing*, INFORMS, v. 32, p. 661–681, Jul. 2020.

Sungur, I.; Ordóñez, F.; Dessouky, M. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, Taylor & Francis Group, v. 40, p. 509–523, Mar. 2008.

Tajik, N. et al. A robust optimization approach for pollution routing problem with pickup and delivery under uncertainty. *Journal of Manufacturing Systems*, Elsevier, v. 33, p. 277–286, 2014.

Toth, P.; Vigo, D. *Vehicle routing : problems, methods, and applications*. 2. ed. [S.l.]: Society for Industrial and Applied Mathematics and the Mathematical Optimization, 2014. 463 p.

Vakili, R.; Shirazi, M. A.; Gitinavard, H. Multi-echelon green open-location-routing problem: A robust-based stochastic optimization approach. *Scientia Iranica*, Sharif University of Technology, v. 28, p. 985–1000, Apr. 2021.

Wang, A.; Subramanyam, A.; Gounaris, C. E. Robust vehicle routing under uncertainty via branch-price-and-cut. *Optimization and Engineering*, Springer, v. 23, p. 1895–1948, Dec. 2022.

Yu, Q.; Cheng, C.; Zhu, N. Robust team orienteering problem with decreasing profits. *INFORMS Journal on Computing*, INFORMS, v. 34, p. 3215–3233, Nov. 2022.

Zang, X. et al. Citespace-based bibliometric review of pickup and delivery problem from 1995 to 2021. *Applied Sciences*, MDPI, v. 12, p. 4607, May 2022.

Appendix A
Dualisation approach

RO approaches seeks to guarantee a feasible solution regardless of the possible data deviations (within the defined uncertainty set). In other words, the methods must provide solutions for the data's worst-case value belonging to the uncertainty set. Finding the worst-case data value means that one must find the either the maximum or minimum values of the parameters. A classical RO approach that consider polyhedral uncertainty sets rely on the strong duality to obtain compact robust counterpart formulations. In this approach, each constraint in which the uncertain parameter appears must contain all of its components. The implementation of this methods commonly result in a formulation with additional variables and constraints which makes the problem harder to be solved (Bertsimas; Sim, 2004).

In the following section we introduce a formulation for the (deterministic) PDPTW which attend the requirements for applying the dualisation technique. This formulation will be used to derive the robust counterpart model using the dualisation approach. Finally, we present its computational results.

A.1 Deterministic formulation

Time window constraints can be ensured by computing the time required to get from one node to another (Chardy; Klopfenstein, 2012). A route solution $u = (u_0 = 0, u_1, ..., u_k = 2n + 1)$ is feasible w.r.t. time windows if the vehicle leaves node i at its time window's opening time, visits all following nodes with no waiting time, and arrives at node j before its time window's closing time, for any pair of nodes i and j such that $0 \leq i < j \leq k$. This approach is presented as follows and will be referred to as *pairwise time window constraints*.

$$e_{u_i} + \sum_{l=i}^{j-1} (d_{u_l} + t_{u_l,u_l+1}) \leq l_{u_j}, \qquad 0 \leq i < j \leq k. \tag{A.1.1}$$

The incorporation of constraints (A.1.1) into MIP formulations for routing variants can be carried out by modelling the path taken by each vehicle. Agra et al. (2012) proposed a layered formulation in which a modified graph is used to incorporate the position of each arc in the route. Their formulation relies on the 3-index VRP formulation (Desaulniers et al., 2002) and results in an additional 4-index binary variable to indicate the position of an arc in a vehicle. The 3-index VRP formulation is known to not perform well on

general-purpose MIP solvers (Toth; Vigo, 2014), in particular for the PDPTW variant (Furtado; Munari; Morabito, 2017).

Based on the multi-commodity flow approach introduced by Garvin et al. (1957), we propose a formulation that incorporates the constraints (A.1.1) in a 2-index compact formulation. To account for the path taken by the vehicle, we introduce a 3-index continuous variable. Let $z_{ijk} = 1$ if the arc $(i, j) \in \mathcal{A}$ is travelled ($x_{ij} = 1$) on the way from the depot 0 to node $k \in \mathcal{N}$, and 0 otherwise. We use the indicator function $[]_\varepsilon$ to denote 1 if the logical expression ε is true, and 0 otherwise.

$$e_i \sum_{k:(i,k)\in\mathcal{A}} z_{ikj} + \sum_{(k,r)\in\mathcal{A}} (d_k + t_{kr})(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) \leq \sum_{h:(j,h)\in\mathcal{A}} x_{jh} l_h, \quad i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\}. \tag{A.1.2}$$

Notice that, instead of ensuring the the time windows at node j, constraints (A.1.11) consider the following node. Variables $z_{i,j,2n+1} = 1$ for $(i, j) \in \mathcal{A}$ such that $x_{ij} = 1$, since all these arcs are travelled in the path from/to the depot. Therefore, we avoid this by considering only customer nodes as j for the pairwise time windows constraints and guaranteeing the time windows at the node that follows j. This can be implemented without loss of generality since the time windows of individual arcs (i, j) is ensured in the arc elimination procedure at the preprocessing stage. Figure 4 illustrates the pairwise time windows constraints (A.1.11).

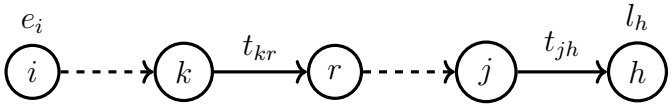


Figure 4 – Illustration of the pairwise time window constraint

The multi-commodity flow formulation for the deterministic PDPTW is proposed as follows.

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \tag{A.1.3}$$

$$\text{s.t.} \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 1, \qquad j \in \mathcal{C} \tag{A.1.4}$$

$$z_{ijk} \leq x_{ij} \qquad (i, j) \in \mathcal{A}, k \in \mathcal{N} \tag{A.1.5}$$

$$\sum_{j:(i,j)\in\mathcal{A}} z_{ijk} = \sum_{j:(j,i)\in\mathcal{A}} z_{jik}, \qquad i \in \mathcal{C}, k \in \mathcal{C} \setminus \{i\} \tag{A.1.6}$$

$$\sum_{j:(0,j)\in\mathcal{A}} z_{0jk} = \sum_{i:(i,k)\in\mathcal{A}} z_{ikk} = 1, \qquad k \in \mathcal{C} \tag{A.1.7}$$

$$\sum_{i:(i,2n+1)\in\mathcal{A}} z_{i,2n+1,k} = \sum_{j:(k,j)\in\mathcal{A}} z_{kjk} = 0, \qquad k \in \mathcal{C} \tag{A.1.8}$$

$$\sum_{j\in\mathcal{N}} z_{i,j,i+n} = 1, \qquad i \in \mathcal{P} \tag{A.1.9}$$

$$\sum_{(i,j)\in\mathcal{A}:j\in\mathcal{C}} z_{ijk}q_j \leq Q, \qquad k \in \mathcal{P} \quad \text{(A.1.10)}$$

$$e_i \sum_{k:(i,k)\in\mathcal{A}} z_{ikj} + \sum_{(k,r)\in\mathcal{A}} (d_k + t_{kr})(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) \leq \sum_{h:(j,h)\in\mathcal{A}} x_{jh}l_h,$$

$$i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\} \quad \text{(A.1.11)}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in \mathcal{A} \quad \text{(A.1.12)}$$

$$z_{ijk} \in \mathbb{R}_+, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{N}. \quad \text{(A.1.13)}$$

The objective function (A.1.3) seeks to minimise the total routing costs. Constraints (A.1.4) ensure that each node is visited exactly once and that the vehicle flow is consistent through the network. Constraints (A.1.5) state bounds for the variable z_{ijk} if a given arc (i,j) is not travelled. The flow balance is preserved along all nodes in constraints (A.1.6). Constraints (A.1.7) sets that there is one arc leaving the initial depot and also one to arrive at the customer. Constraints (A.1.8) sets that there is no incoming arc to the final depot that is used to arrive at customer nodes. The precedence between pickup and delivery nodes is ensured by constraints (A.1.9). Vehicle capacity is ensured by constraints (A.1.10). Constraints (A.1.11) are the pairwise time windows constraints. The domain of all variables are defined in (A.1.12) and (A.1.13). This formulation can also model the VRPTW by dropping the precedence constraints (A.1.9).

An alternative set of constraints for a multi-commodity flow formulation was also presented by Jaillet, Qi and Sim (2016), the linear decision rule formulation. In preliminary experiments, the model presented above showed superior performance compared with the linear decision rule model. Some straightforward variable fixing are possible in this formulation, such as $z_{ij0} = 0$ for all $(i,j) \in \mathcal{A}$, $z_{ijk} = 0$ for all $j \in \mathcal{N}$ if $e_i + d_i + t_{ik} < l_k$. For liftings on multi-commodity flow formulation the reader is referred to the work of Letchford and Salazar-González (2015) which proposes models for the CVRP.

A.2 Robust counterpart formulation

The dualisation scheme for obtaining robust counterpart formulations is applied to the capacity and time windows constraints (A.1.10) and (A.1.11), respectively. In both cases, each constraint contains all components of the uncertain parameter.

To guarantee the feasibility of the capacity constraints (A.1.10) regarding all demand realisation belonging to the uncertainty set (3.1.1) we must ensure the following constraints.

$$\sum_{(i,j)\in\mathcal{A}:j\in\mathcal{C}} z_{ijk}\tilde{q}_j \leq Q, \qquad k \in \mathcal{P}, \tilde{q} \in \mathcal{U}^q. \quad \text{(A.2.1)}$$

Note that these it could result in a large number of constraints due to the enumeration of all demand realisations in the uncertainty set. Instead, we ensure the capacity

robustness by incorporating the maximisation of the worst-case of demand realisation.

$$\max_{\tilde{q}\in\mathcal{U}^q}\left\{\sum_{(i,j)\in\mathcal{A}:j\in\mathcal{C}} z_{ijk}\tilde{q}_j\right\} \leq Q, \qquad k\in\mathcal{P}. \tag{A.2.2}$$

From the definition of the uncertainty set (3.1.1) we see that the demand realisation $\tilde{q}_i$ consists in a linear combination of its nominal demand $\bar{q}_i$ and demand deviation $\xi_i^q\hat{q}_i$. Therefore, we can separate the nominal part of the demand realisation from the maximisation problem. Additionally, we reformulate the objective function's summation since the deviations ξ^q are modelled for all $i\in\mathcal{P}$.

$$\sum_{(i,j)\in\mathcal{A}:j\in\mathcal{C}} z_{ijk}\bar{q}_j + \max_{\substack{\xi^q\in[0,1]^n\\ \sum\xi^q\leq\Gamma^q}}\left\{\sum_{(i,j)\in\mathcal{A}:j\in\mathcal{P}} z_{ijk}(\xi_j^q\hat{q}_j + \xi_{j+n}^q\hat{q}_{j+n})\right\} \leq Q, \qquad k\in\mathcal{P}. \tag{A.2.3}$$

Since the inner maximization problem is feasible and bounded, by strong duality, its dual is also feasible and bounded and their objective values coincide. Let $\eta_{jk}\in\mathbb{R}_+$ be the dual variables associated with constraints $\xi^q\in[0,1]^n$ and $\mu_k\in\mathbb{R}_+$ be the dual variables associated with constraints $\sum\xi^q\leq\Gamma^q$. Then, the dual formulation of the inner maximisation problem in (A.2.3) is presented as follows:

$$\min \quad \sum_{j\in\mathcal{P}}\eta_{jk} + \Gamma^q\mu_k \tag{A.2.4}$$

$$\text{s.t.} \quad \eta_{jk}+\mu_k \geq \sum_{i:(i,j)\in\mathcal{A}}(\hat{q}_j z_{ijk} + \hat{q}_{j+n}z_{i,j+n,k}) \qquad j,k\in\mathcal{P} \tag{A.2.5}$$

$$\eta_{jk}\in\mathbb{R}_+, \qquad j\in\mathcal{P}, k\in\mathcal{P} \tag{A.2.6}$$

$$\mu_k\in\mathbb{R}_+, \qquad k\in\mathcal{P}. \tag{A.2.7}$$

This formulation can replace the inner maximisation problem in constraints (A.2.3). Since a feasible solution for problem (A.2.4)-(A.2.7) is enough to ensure robustness, given the less-or-equal constraint, we drop the minimisation objective since. The robust capacity constraints (A.2.2) are rewritten as follows.

$$\sum_{(i,j)\in\mathcal{A}} z_{ijk}\bar{q}_j + \sum_{j\in\mathcal{P}}\eta_{jk} + \Gamma^q\mu_k \leq Q, \qquad k\in\mathcal{P} \tag{A.2.8}$$

$$\eta_{jk}+\mu_k \geq \sum_{i:(i,j)\in\mathcal{A}}(\hat{q}_j z_{ijk} + \hat{q}_{j+n}z_{i,j+n,k}), \qquad j,k\in\mathcal{P} \tag{A.2.9}$$

$$\eta_{jk}\in\mathbb{R}_+, \qquad j\in\mathcal{P}, k\in\mathcal{P} \tag{A.2.10}$$

$$\mu_k\in\mathbb{R}_+, \qquad k\in\mathcal{P}. \tag{A.2.11}$$

For the robustness of the time windows constraints (A.1.11), we follow similar steps. The constraint must be valid for all travel time realisations within the uncertainty set (3.1.2).

$$e_i\sum_{k:(i,k)\in\mathcal{A}} z_{ikj} + \sum_{(k,r)\in\mathcal{A}}(d_k+\tilde{t}_{kr})(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) \leq \sum_{h:(j,h)\in\mathcal{A}} x_{jh}l_h,$$

$$i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\}, \tilde{t} \in \mathcal{U}^t. \quad \text{(A.2.12)}$$

As in the demand uncertainty case, it leads to an exponential number of constraints depending on the size of n and Γ^t. To seize feasibility in the worst-case, we incorporate an internal maximisation problem. We separate the constant (nominal) part of the travel time realisation from the maximisation problem. Then, the robust counterpart constraints are as follows.

$$e_i \sum_{k:(i,k)\in\mathcal{A}} z_{ikj} + \sum_{(k,r)\in\mathcal{A}} (d_k + \bar{t}_{kr})(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) + \max_{\substack{\xi^t \in [0,1]^{|\mathcal{A}|} \\ \sum \xi^t \leq \Gamma^t}} \left\{ \sum_{(k,r)\in\mathcal{A}} \xi^t_{kr} \hat{t}_{kr}(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) \right\} \leq \sum_{h:(j,h)\in\mathcal{A}} x_{jh} l_h, \qquad i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\}. \quad \text{(A.2.13)}$$

However, the problem is still hard to solve due to the inner maximisation problem. As in the dualisation scheme, one can transform the internal problem into its dual formulation and consequently drop minimisation objective since, if the optimal solution of the dual problem satisfies the constraint feasibility, a dual feasible solution also satisfies. Then, carrying out the same steps as done previously for the demand uncertainty, the following set of new constraints are linear.

$$e_i \sum_{k:(i,k)\in\mathcal{A}} z_{ikj} + \sum_{(k,r)\in\mathcal{A}} (d_k + \bar{t}_{kr})(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) + \sum_{(k,r)\in\mathcal{A}} \lambda_{krij} + \Gamma^t \rho_{ij} \leq \sum_{h:(j,h)\in\mathcal{A}} x_{jh} l_h, \quad i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\} \quad \text{(A.2.14)}$$

$$\lambda_{krij} + \rho_{ij} \geq \hat{t}_{kr}(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) \qquad (k,r) \in \mathcal{A}, i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\}. \quad \text{(A.2.15)}$$

where $\lambda_{krij} \in \mathbb{R}_+$ are the dual variables associated with constraints $\xi^t \in [0,1]^{|\mathcal{A}|}$ and $\rho_{ij} \in \mathbb{R}_+$ are the dual variables associated with constraints $\sum \xi^t \leq \Gamma^t$.

Then, obtain the robust counterpart formulation that handles demand and travel time uncertainties considering their realisations belonging to sets (3.1.1) and (3.1.2).

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}, \quad \text{(A.2.16)}$$

$$\text{s.t.} \quad \text{(A.1.4)} - \text{(A.1.9)}, \text{(A.1.12)} - \text{(A.1.13)}$$

$$\sum_{(i,j)\in\mathcal{A}} z_{ijk} \bar{q}_j + \sum_{j\in\mathcal{P}} \eta_{jk} + \Gamma^q \mu_k \leq Q, \qquad k \in \mathcal{P} \quad \text{(A.2.17)}$$

$$\eta_{jk} + \mu_k \geq \sum_{i:(i,j)\in\mathcal{A}} (\hat{q}_j z_{ijk} + \hat{q}_{j+n} z_{i,j+n,k}), \qquad j,k \in \mathcal{P} \quad \text{(A.2.18)}$$

$$e_i \sum_{k:(i,k)\in\mathcal{A}} z_{ikj} + \sum_{(k,r)\in\mathcal{A}} (d_k + \bar{t}_{kr})(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) + \sum_{(k,r)\in\mathcal{A}} \lambda_{krij} + \Gamma^t \rho_{ij} \leq \sum_{h:(j,h)\in\mathcal{A}} x_{jh} l_h, \qquad i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\} \quad \text{(A.2.19)}$$

$$\lambda_{krij} + \rho_{ij} \geq \hat{t}_{kr}(z_{krj} - z_{kri} + x_{jr}[]_{k=j}) \qquad (k,r) \in \mathcal{A}, i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\} \quad \text{(A.2.20)}$$

$$\eta_{jk} \in \mathbb{R}_+, \qquad j \in \mathcal{P}, k \in \mathcal{P} \quad \text{(A.2.21)}$$

$$\mu_k \in \mathbb{R}_+, \qquad k \in \mathcal{P} \quad \text{(A.2.22)}$$

$$\lambda_{krij} \in \mathbb{R}_+, \qquad (k,r) \in \mathcal{A}, i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\} \quad \text{(A.2.23)}$$

$$\rho_{ij} \in \mathbb{R}_+, \qquad i \in \mathcal{N}, j \in \mathcal{C} \setminus \{i\}. \quad \text{(A.2.24)}$$

The objective function (A.2.16) minimises the total routing cost. From the formulation for the deterministic PDPTW we maintain the degree constraints (A.1.4), the commodity-flow constraints (A.1.5) to (A.1.8), the precedence constraints (A.1.9), and the domain of the variables (A.1.12) and (A.1.13). Constraints (A.2.17) are the capacity constraints that guarantee feasibility for all demand realisations within the uncertainty set. Constraints (A.2.18)-(A.2.22) are the the dual constraints and variables domain related to the capacity constraints. Constraints (A.2.19) are the time window constraints that ensure feasibility for all travel time realisation within the uncertainty set. Constraints (A.2.20)-(A.2.24) are the dual constraints and variables domain related to the time windows constraints.

A.3 Computational performance

We assess the performance of the proposed approach through extensive computational experiments. All experiments were performed on a Linux PC with an Intel Core i7-12700 2.10 GHz processor and 16 GB of RAM, and using the general-purpose optimisation solver Gurobi version 11.0 with its default parameters. The formulation was implemented in C++ and the time limit for each execution was set to 1 hour (3600 seconds). We consider the same benchmark instances as described in Section 5.1. The formulation (A.2.16)-(A.2.24) was directly implemented using the Gurobi solver. The instances and the results are also available in the supplementary material at <https://github.com/abreualexp/master-results>.

Table 10 presents a summary of the computational performance when addressing different sources of uncertainty. The results are the percentage of instances that have been solved to optimality, have stopped due to the time limit, or have run out of memory. More precisely, the table reports the percentages of instances for which proven optimal solutions were found within 3600 seconds (*Optimal*), the time limit was reached (*Time limit*), and the memory limit was reached (*Memory limit*). The first row presents the results for the deterministic instances, while the subsequent three rows correspond to uncertainty in demands, travel times, and both. The final row provides the average across all instances. The proportion of instances solved to optimality is about 28%. While the method found optimal solutions for over 35% in configurations with no uncertainty and demand uncertainty, less than 25% of instances were optimally solved in the presence of travel time uncertainty. On the other hand, the time limit was reached more often

in instances with no uncertainty and demand uncertainty, namely about 17% and 15%, respectively. Then, the proportion of instances in which the method reached memory limit is higher in instances with travel time instances. In fact, the proportion of instances with travel time uncertainty that reached memory limit is 75% on average. In general, the method reached memory limit in over 65% of the instances.

Uncertainty		Optimal (%)	Time limit (%)	Memory limit (%)
Demand	Travel time			
		35.0	16.7	48.3
✓		36.0	15.2	48.9
	✓	22.3	2.1	75.6
✓	✓	24.4	1.1	74.4
Overall average		27.9	6.5	65.6

Table 10 – Summary of instance solution status among different uncertainty configuration for the dualisation approach.

In Table 11, we analyse the scalability of both formulations across instances of varying sizes. The table presents the average results for each approach, grouped by the number of requests. Specifically, the table shows the proportion of optimal solutions (*Opt.*); the average elapsed computation time to solve instances to optimality (*Time*); and the average optimality gap for instances that reached the time or memory limit (*Gap*), as provided by the solver. The method found optimal solutions for all instances with up to 15 requests. However, it stops finding optimal solutions for instances with at least 45 requests. From 20 to 40 requests, the amount of optimal solutions found reduce rapidly. The computation time required to find optimal solutions increase as the number of request grows, with the only exception of instances with 30 requests. Also, the average optimality gap starts with over 55% in instances with 20 requests and increases reaching about 95% in instances with 75 requests. In general, as the size of the instances grow, the method finds less optimal solutions, takes longer times to find them, and returns higher gaps if it fails to find it within the time or memory limit.

In Table 12, we analyse how the formulation performs under different uncertainty settings across each instance class (AA, BB, CC, and DD). The table presents the same metrics as those shown in Table 11. For each uncertainty setting, we observe that the proportion of optimal solutions decreases gradually as instance classes progress from AA to DD. With the only exception of instances of the BB class in which the method found more optimal solutions in some uncertainty configurations compared with instances of the AA class. Notably, increasing the time window length has a more significant impact on the proportion of optimal solution found compared with increasing capacity. Among each instance class, we observe that less optimal solutions are found for instances with only travel time uncertainty. Considering instances with only travel time uncertainty, the

n	Optimal (%)	Time (s)	Gap (%)
5	100.0	< 0.1	
10	100.0	2.7	
15	100.0	177.7	
20	71.4	263.6	55.7
25	33.0	665.3	65.0
30	19.6	195.9	71.7
35	15.2	714.8	74.3
40	1.8	3149.3	69.8
45	0.0		57.6
50	0.0		66.3
55	0.0		73.4
60	0.0		75.9
65	0.0		93.0
70	0.0		93.9
75	0.0		94.6

Table 11 – Average results among the different instance sizes for the dualisation approach.

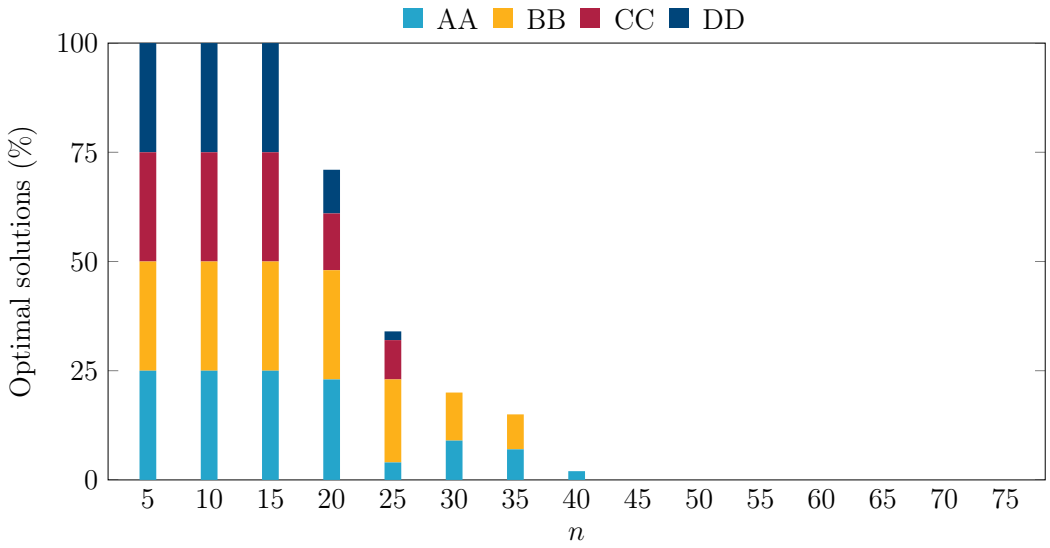
computation time is smaller in some cases. We note no further common behaviour for the results of average computation time. In general, the average optimality gaps are over 50% for any uncertainty configuration and instance class.

Class	Uncertainty		Optimal (%)	Time (s)	Gap (%)
	Demand	Travel time			
AA			40.0	548.7	56.9
	✓		40.2	248.0	55.0
		✓	25.0	27.6	75.9
	✓	✓	25.0	129.6	77.4
BB			40.0	91.0	66.9
	✓		45.5	201.7	69.6
		✓	26.5	16.9	75.9
	✓	✓	33.3	32.3	51.1
CC			33.3	249.9	83.8
	✓		31.8	278.7	82.2
		✓	19.7	199.2	81.4
	✓	✓	20.5	209.4	77.7
DD			26.7	470.3	81.2
	✓		26.5	182.2	77.1
		✓	18.2	630.0	86.7
	✓	✓	18.9	208.6	84.0

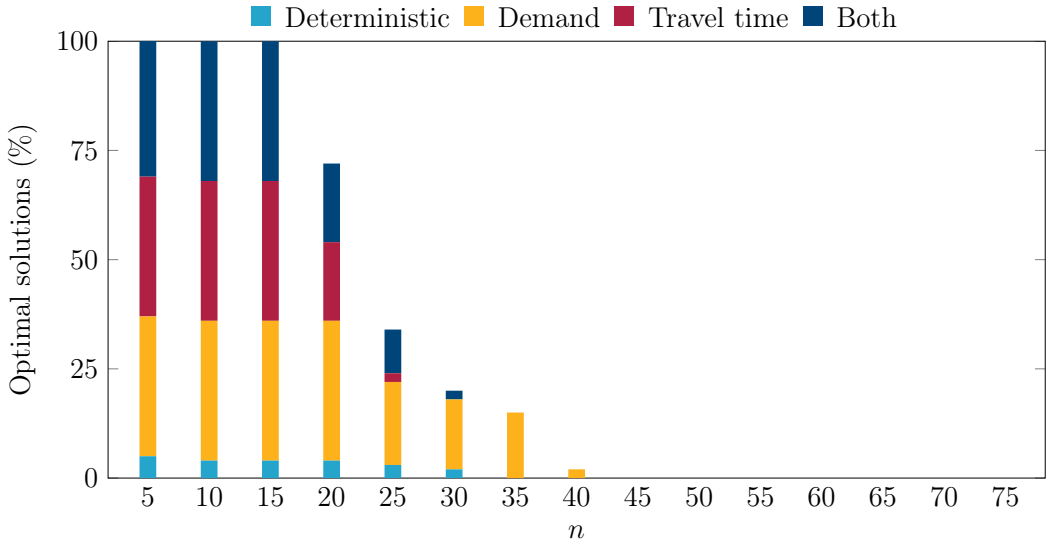
Table 12 – Average results among instance classes and uncertainty configurations for the dualisation approach.

Figures 5a and 5b illustrate the scalability of optimal solutions found, measured by

the number of requests. In Figure 5a, results are grouped into classes, with the stacked bars (from bottom to top) corresponding to classes AA, BB, CC, and DD. In Figure 5b, results are grouped by uncertainty configuration, where the stacked bars (from bottom to top) represent deterministic instances (no uncertainty), uncertain demands, uncertain travel times, and both uncertainties.



(a) Optimal solutions for each instance class.



(b) Optimal solutions for each uncertainty configuration.

Figure 5 – Optimal solutions (in percentage) among different instance sizes and characteristics for the dualisation approach.

In Table 13, we analyse how the formulation behave under different uncertainty levels. The first two columns present the various combinations of uncertainty budgets. In the first row, results for deterministic instances ($\Gamma^q = \Gamma^t = 0$) are shown. The subsequent rows display results for the three levels of the uncertainty budget (1, 5, and 10). Rows two to four correspond to the demand uncertainty configuration, while rows five to seven refer

to travel time uncertainty. Results for instances with both uncertainties are presented in the final three rows. We note that as the level of uncertainty increases, the method finds less optimal solutions with the only exception of instances with only demand uncertainty. In this uncertainty configuration, the optimal solutions increase from levels 1 to 5 and decrease in levels 5 to 10.

Γ^q	Γ^t	Optimal (%)	Time (s)	Gap (%)
0	0	35.0	331.9	14.9
1	0	36.1	148.2	15.1
5	0	38.3	278.6	15.1
10	0	33.3	258.5	15.1
0	1	25.0	256.9	16.8
0	5	23.3	132.2	16.8
0	10	18.5	151.2	16.8
1	1	29.4	247.0	17.0
5	5	24.4	38.0	16.8
10	10	19.0	56.5	16.8

Table 13 – Average results among different levels of uncertainty for the dualisation approach.