The robust pickup and delivery problem with time windows

Alex Abreu[a], Maria Battarra[b,*], Luciano Costa[c], Pedro Munari[a]

[a] *Federal University of São Carlos, Production Engineering Department, São Carlos, SP, 13565-905, Brazil*
[b] *University of Bath, School of Management, Bath, BA2 7AY, United Kingdom*
[c] *Federal University of Paraíba, Production Engineering Department, João Pessoa, PB, 58051-900, Brazil*

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

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 et al., 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).

*Corresponding author
Email addresses: abreualexp@gmail.com (Alex Abreu), mb2182@bath.ac.uk (Maria Battarra), luciano.costa@academico.ufpb.br (Luciano Costa), munari@dep.ufscar.br (Pedro Munari)

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 et al., 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 et al., 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 et al., 2020; Wang et al., 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 and Sim, 2004).

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). In our formulation, both demands and travel times are considered uncertain and are modelled using the cardinality-constrained uncertainty set. The main contributions of this article are as follows:

- We propose two robust counterpart formulations to ensure robustness for capacity and time windows. The first approach is a compact formulation, i.e., a model with polynomial numbers of variables and constraints. The second formulation relies on cutting planes, resulting in a model 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.

The remainder of this paper is organised as follows. Section 2 reviews the literature related to our study. In Section 3, we formally define the deterministic and robust problems. The proposed robust counterpart formulations are presented in Section 4. Section 5 details the B&C algorithm, including its configuration and the separation procedures. The experimental design is described in Section 6, along with the computational results and robustness analysis. Finally, Section 7 summarises our conclusions and outlines opportunities for future research.

2. Related literature

Deterministic variants of PDPs have been extensively addressed in the literature, encompassing different demand configurations, namely the 1-M-1, M-M, and 1-1 variants (Berbeglia et al., 2007; Parragh et al., 2008a,b). While many studies propose heuristic procedures for the PDPTW and its variants, the literature has primarily focused on advancing exact methods for this problem using compact formulations, B&C algorithms, and branch-and-price algorithms (Ropke et al., 2007; Baldacci et al., 2011; Furtado et al., 2017). Readers interested in surveys on PDP variants are referred to Çağrı Koç and Laporte (2018) for 1-M-1 problems; Çağrı Koç et al. (2020) for the PDP with simultaneous pickup and delivery; Cordeau et al. (2008) for 1-1 problems; Berbeglia et al. (2010) for dynamic problems; and Battarra et al. (2014) for other variants.

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.

Other studies have addressed different PDP variants using RO approaches, primarily focusing on rich 1-M-1 and M-M problems. The 1-M-1 structure involves transporting goods from the depot to customers and from suppliers to the depot. 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. Al Chami et al. (2018) and Al 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.

Tajik et al. (2014), who first applied RO techniques to a PDP, addressed a 1-M-1 variant with soft time windows, backhaul constraints, and pollution costs as the objective function. They considered service and travel times, fuel consumption, and greenhouse gas emission costs as uncertain parameters, modelling them using bounded box uncertainty sets. Santos et al. (2020) focused on profit maximisation with selective demands and backhaul constraints. They modelled

pickup revenue as an uncertain parameter, using cardinality-constrained and factor model uncertainty sets. Other studies have incorporated additional operational practices or considerations into 1-M-1 variants, such as: cross-docks (Mousavi and Vahdani, 2017), production and inventory (Golsefidi and Jokar, 2020), multi-product operations (Mondal and Roy, 2021), workforce allocation (Pavlova and Shichiyakh, 2021), environmental considerations (Vakili et al., 2021), automated vehicle platoons (Pourmohammad-Zia et al., 2023), and multi-depot operations (Guo et al., 2024).

The M-M PDP variants are characterised by the transportation of a commodity with multiple origins and destinations. Huang et al. (2020) modelled freight shipping services between regions separated by an ocean, specifically motivated by a hub-and-spoke maritime network application. They considered demands and costs as uncertain parameters, defined within a discrete scenario-based uncertainty set. Allahyari et al. (2021) addressed the transportation of valuable goods, where security carriers aim to minimise both robbery risk and operational costs. They modelled uncertain demands using a hard worst-case approach, where all uncertain parameters simultaneously assume their worst-case values, resulting in an overly conservative solution with a poor objective function. Hansuwa et al. (2022) studied simultaneous pickup and delivery with hard time windows. They modelled uncertain demands, travel times, and service times using box and ellipsoidal uncertainty sets. For the box uncertainty set, they derived linear formulations through dualisation techniques and the linearisation of recursive equations.

RO approaches have also been applied in the context of other VRP variants, employing different techniques to derive tractable robust counterpart methods. The RVRP under demand uncertainty was studied by Gounaris et al. (2013) and Pessoa et al. (2021). Moreover, several publications address the robust *VRP with time windows* (RVRPTW) under travel time uncertainty (Agra et al., 2012, 2013), demand uncertainty (Lu and Gzara, 2019), and both demand and travel time uncertainties (Munari et al., 2019; Wang et al., 2022; Campos et al., 2024). The VRP with deadlines under demand and travel time uncertainty was also considered by Lee et al. (2012). While compact robust formulations are commonly derived using the dualisation technique, several authors have utilised the linearisation of recursive equations to obtain robust counterparts for VRP variants (Munari et al., 2019; De La Vega et al., 2020; Campos et al., 2024). They offer the advantage of being explicitly incorporated in vehicle flow formulations.

We rely on the advances of RO approaches for VRP variants to introduce RO methods to the RPDPTW. Both demands and travel times are uncertain and are modelled using the cardinality-constrained uncertainty set. We propose a compact robust counterpart formulation, obtained through linearisation, and a tailored B&C algorithm.

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 et al., 1991; Ropke et al., 2007; Ropke and 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 $\xi_i^q, \xi_{ij}^t \in [-1,1]$ be the normalised scale deviations of demand and travel time, respectively. They indicate the level of deviation from their nominal values, i.e., $\xi_i^q = (\tilde{o}_i - \bar{o}_i)/\hat{o}_i$ and $\xi_{ij}^t = (\tilde{t}_{ij} - \bar{t}_{ij})/\hat{t}_{ij}$. The demand and travel time realisations can then be expressed as a linear combination of their nominal and deviation components: $\tilde{o}_i = \bar{o}_i + \xi_i^q \hat{o}_i$ and $\tilde{t}_{ij} = \bar{t}_{ij} + \xi_{ij}^t \hat{t}_{ij}$. 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_i^q 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 and Sim, 2004). 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:

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

$$\mathcal{U}^t = \left\{ \tilde{t} \in \mathbb{R}_+^{|\mathcal{A}|} : \tilde{t}_{ij} = \bar{t}_{ij} + \xi_{ij}^t \hat{t}_{ij}, (i,j) \in \mathcal{A}; \sum_{(i,j) \in \mathcal{A}} \xi_{ij}^t \leq \Gamma^t; \xi_{ij}^t \in [0,1]^{|\mathcal{A}|} \right\}. \quad (2)$$

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 et al., 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:

$$\mathcal{W}_{u_j\gamma} = \left\{ u_i \in \mathcal{P} : Q_{u_{i-1}\gamma-1} + \hat{q}_{u_i} > Q_{u_{i-1}\gamma}, i = 1, \ldots, j \right\}, \qquad j = 0, \ldots, k, \gamma = 1, \ldots, \Gamma^q. \quad (3)$$

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

$$Q_{u_j\gamma} = \begin{cases} 0, & \text{if } j = 0, & (4a) \\ Q_{u_{j-1}\gamma} + \bar{q}_{u_j}, & \text{if } u_j \in \mathcal{P} \wedge \gamma = 0, & (4b) \\ \max\{Q_{u_{j-1}\gamma} + \bar{q}_{u_j},\ Q_{u_{j-1},\gamma-1} + \bar{q}_{u_j} + \hat{q}_{u_j}\}, & \text{if } u_j \in \mathcal{P} \wedge \gamma > 0, & (4c) \\ Q_{u_{j-1},\gamma} + \bar{q}_{u_j}, & \text{if } u_j \in \mathcal{D} \wedge u_j - n \notin \mathcal{W}_{u_{j-1},\gamma}, & (4d) \\ Q_{u_{j-1},\gamma} + \bar{q}_{u_j} + \hat{q}_{u_j}, & \text{if } u_j \in \mathcal{D} \wedge u_j - n \in \mathcal{W}_{u_{j-1},\gamma}. & (4e) \end{cases}$$

Cases (4a)–(4c) are analogous to the recursive equations stated for the RVRPTW (Munari et al., 2019), but are applied exclusively to pickup nodes. Case (4a) represents the boundary condition, ensuring that there is no vehicle load after leaving the depot. In Case (4b), 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 (4c) 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 (4c). Finally, the pairing property between the pickup and delivery nodes of a request is ensured by matching their demand realisations in cases (4d) and (4e).

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 et al., 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:

$$Q_h = \sum_{i \in \mathcal{S}_h^+} \bar{q}_i + \max_{\substack{\mathcal{Q} \subseteq \mathcal{S}_h^+ \\ |\mathcal{Q}| \leq \Gamma}} \sum_{i \in \mathcal{Q}} \hat{q}_i \qquad 1 \leq h \leq k. \tag{5}$$

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.

□

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:

$$B_{u_j\gamma} = \begin{cases} e_{u_j}, & \text{if } j = 0, \quad (6a) \\ \max\{e_{u_j}, B_{u_{j-1}\gamma} + d_{u_{j-1}} + \bar{t}_{u_{j-1},u_j}\}, & \text{if } \gamma = 0, \quad (6b) \\ \max\{e_{u_j}, B_{u_{j-1},\gamma} + d_{u_{j-1}} + \bar{t}_{u_{j-1},u_j}, & \\ \qquad B_{u_{j-1},\gamma-1} + d_{u_{j-1}} + \bar{t}_{u_{j-1},u_j} + \hat{t}_{u_{j-1},u_j}\}, & \text{otherwise.} \quad (6c) \end{cases}$$

Case (6a) represents the boundary condition at the depot, stating that the service at the depot begins at its time window's opening time. Case (6b) 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 (6c) 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., $B_{u_j\Gamma^t} \leq l_{u_j}$.

4. Robust counterpart formulations

In this study, the formulation proposed by Furtado et al. (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.

We adopt the same notation as in Section 3. 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:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{7}$$

$$\text{s.t.} \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 1, \qquad j \in \mathcal{C} \tag{8}$$

$$Q_j \geq Q_i + q_j - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} \tag{9}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\}, \qquad i \in \mathcal{N} \tag{10}$$

$$B_j \geq B_i + t_{ij} + d_i - (l_i - e_j)(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} \tag{11}$$

$$e_i \leq B_i \leq l_i, \qquad i \in \mathcal{N} \tag{12}$$

$$B_{n+i} \geq B_i + t_{i,n+i}, \qquad i \in \mathcal{P} \tag{13}$$

$$v_{n+i} = v_i, \qquad i \in \mathcal{P} \tag{14}$$

$$jx_{0j} \leq v_j \leq jx_{0j} - n(x_{0j} - 1), \qquad j \in \mathcal{C} \tag{15}$$

$$n(x_{ij} - 1) \leq v_j - v_i \leq n(1 - x_{ij}), \qquad i,j \in \mathcal{C} : i \neq j \tag{16}$$

$$1 \leq v_j \leq n, \qquad j \in \mathcal{C} \tag{17}$$

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in \mathcal{A}. \tag{18}$$

The objective function (7) seeks to minimise the total routing costs. Constraints (8) ensure that each customer node is visited exactly once and that the vehicle flow is consistent throughout the network. Constraints (9) propagate the load on the vehicle travelling between two nodes, while constraints (10) ensure that the vehicle's capacity is not exceeded. Constraints (11) and (12) propagate the time flow, prevent subtours, and ensure that time windows are respected. Constraints (13)–(16) guarantee the precedence and pairing relationships between the pickup and delivery nodes of a request. The domains of the variables are defined in (10), (12), (17), and (18).

From formulation (7)–(18), 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 (1) and (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 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 et al., 2013; De La Vega et al., 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 et al., 2024).

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 (4). 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), within the formulation. Thus, the recursive equation (4) can be reformulated as the following set of linear constraints:

$$\max\{0, \bar{q}_j\} \leq Q_{j\gamma} \leq \min\{Q, Q + \bar{q}_j\}, \qquad j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^q \tag{19}$$

$$Q_{j\gamma} \geq Q_{i\gamma} + \bar{q}_j x_{ij} - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 0, \ldots, \Gamma^q \tag{20}$$

$$Q_{j\gamma} \geq Q_{i,\gamma-1} + (\bar{q}_j + \hat{q}_j) x_{ij} - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 1, \ldots, \Gamma^q \tag{21}$$

$$Q_{j\gamma} \geq Q_{i\gamma} + \bar{q}_j x_{ij} + \hat{q}_j W_{i,\gamma,j-n} - Q(1 - x_{ij}), \qquad (i,j) \in \mathcal{A} : j \in \mathcal{D}, \gamma = 0, \ldots, \Gamma^q. \tag{22}$$

The domain of $Q_{j\gamma}$ is defined by constraints (19), which ensure that the vehicle capacity is respected at each node for any level of uncertainty. Constraints (20) and (21) jointly correspond to cases (4b) and (4c), 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 (20) ensure that the load $Q_{j\gamma}$ increases by at least the nominal demand of j, while constraints (21) also account for the customer demand deviation when the number of requests in their worst-case (γ) increases by one unit. Constraints (22) 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) 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:

$$W_{j\gamma j} \geq \frac{Q_{i,\gamma-1} + \hat{q}_j - Q_{i\gamma}}{\hat{q}_j + \max\limits_{i\in\mathcal{P}} \hat{q}_i} + x_{ij} - 1, \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 1, \ldots, \Gamma^q \quad (23)$$

$$W_{j\gamma j} \geq W_{i,\gamma-1,j} + \frac{Q_{i,\gamma-1} + \hat{q}_j - Q_{i\gamma}}{\hat{q}_j + \max\limits_{i\in\mathcal{P}} \hat{q}_i} + x_{ij} - 1, \qquad (i,j) \in \mathcal{A} : j \in \mathcal{P}, \gamma = 1, \ldots, \Gamma^q \quad (24)$$

$$W_{j\gamma h} \geq W_{i\gamma h} + x_{ij} - 1, \qquad (i,j) \in \mathcal{A}, h \in \mathcal{P}, \gamma = 0, \ldots, \Gamma^q \quad (25)$$

$$W_{j\gamma h} \in \{0,1\}, \qquad j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^q, h \in \mathcal{P}. \quad (26)$$

Constraints (23) ensure that $W_{j\gamma j} = 1$ if the demand of the pickup node j attains its worst-case. Constraints (24) 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 (25) 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 (26). It is worth mentioning that, to the best of our knowledge, the variables $W_{j\gamma h}$ and the linearized constraints (19)–(26) are introduced in this paper, as they are specifically developed to account for paired pickup and delivery operations.

The linearisation of the recursive equation (6), 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:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \quad (27)$$

$$\text{s.t.} \quad (8), (14) - (26)$$

$$B_{j\gamma} \geq B_{i\gamma} + d_i + \bar{t}_{ij} - (l_i - e_j)(1 - x_{ij}), \qquad (i,j) \in \mathcal{A}, \gamma = 0, \ldots, \Gamma^t \quad (28)$$

$$B_{j\gamma} \geq B_{i,\gamma-1} + d_i + \bar{t}_{ij} + \hat{t}_{ij} - (l_i - e_j)(1 - x_{ij}), \qquad (i,j) \in \mathcal{A}, \gamma = 1, \ldots, \Gamma^t \quad (29)$$

$$B_{n+j,\gamma} \geq B_{j\gamma} + d_j + \bar{t}_{j,n+j}, \qquad j \in \mathcal{P}, \gamma = 0, \ldots, \Gamma^t \quad (30)$$

$$e_j \leq B_{j\gamma} \leq l_j, \qquad j \in \mathcal{N}, \gamma = 0, \ldots, \Gamma^t. \quad (31)$$

The objective function (27) seeks to minimise the total routing cost. Constraints (8) (degree constraints), (14)–(16) (pairing constraints), and (17) and (18) (variable domains) are similar to those in the deterministic case. Capacity robustness under demand uncertainty is ensured by constraints (19)–(26). Time window robustness under travel time uncertainty is ensured by constraints (28)–(31). Constraints (28) and (29) function as cases (6a) and (6b), 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 (30). Constraints (31) 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 (8)–(18), 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 (9) and (10), and time window constraints (11) and (12), 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 et al., 2013; Campos et al., 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 (32), 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.

$$x(\delta^+(\mathcal{S})) \geq \max\left\{1, \left\lceil \frac{r(\mathcal{S})}{Q} \right\rceil\right\}, \qquad \mathcal{S} \subseteq \mathcal{C} : |\mathcal{S}| \geq 2. \tag{32}$$

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:

$$r(\mathcal{S}) = \max\left\{\sum_{i\in\mathcal{S}^+} \bar{q}_i + \max_{\substack{\mathcal{Q}\subseteq\mathcal{S}^+ \\ |\mathcal{Q}|\leq\Gamma^q}} \sum_{i\in\mathcal{Q}} \hat{q}_i, \left| \sum_{i\in\mathcal{S}^-} \bar{q}_i + \min_{\substack{\mathcal{Q}\subseteq\mathcal{S}^- \\ |\mathcal{Q}|\leq\Gamma^q}} \sum_{i\in\mathcal{Q}} \hat{q}_i, \right|\right\}. \tag{33}$$

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 et al., 2024). We implemented a strengthened version of this inequality, the tournament constraints of Ascheuer et al. (2000), as also done in other robust routing variants (Wang et al., 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 et al., 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:

$$\min \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij} \tag{34}$$

s.t. (8) – (18),

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

$$\mathcal{S} \subseteq \mathcal{C} : |\mathcal{S}| \geq 2 \tag{35}$$

$$\sum_{i=1}^{k-1} \sum_{j \in \mathcal{L}_i(u)} x_{u_i, u_j} \leq k - 2, \qquad u \in \mathcal{I}. \tag{36}$$

This formulation seeks to minimise the total routing cost (34) while satisfying the degree constraints (8), capacity constraints (9)–(10), time window constraints (11)–(12), precedence constraints (13), and pairing constraints (14)–(16), along with the variable domains (17)–(18). Furthermore, constraints (35) 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 (36) 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 5, 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.

5. Branch-and-cut

The B&C algorithm dynamically incorporates RRCIs (35) and RIPECs (36) 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.

5.1. Robust rounded capacity inequalities

The algorithm for separating RRCIs (35) 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 1 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 1: 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 $\mathcal{S}$.

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

5.2. 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 (36) 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 (6) 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 2 describes the structures used in the algorithm.

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 2: Description of the algorithm's notation for separating RIPECs.

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 (described in Section 3.1) 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 (6) 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 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).

Algorithm 2: Separation of RIPECs

```
1  function SeparationRIPEC(x)
2      B_{jγ} ← 0, for all j ∈ N, γ = 0, ..., Γ^t
3      E ← ∅
4      foreach h ∈ C do
5          if h ∈ E then
6              Continue to next h
7          if x_{0h} > 0 then
8              Set u = (0, h) and k ← 2
9              B_{h0} ← max{e_h, e_0 + t̄_{0h}}
10             B_{hγ} ← max{e_h, e_0 + t̄_{0h} + t̂_{0h}}, for all γ = 1, ..., Γ^t
11         else
12             Set u = (h) and k ← 1
13             B_{hγ} ← e_h, for all γ = 0, ..., Γ^t
14         feasible_path ← true
15         j ← h
16         while j ≠ 2n + 1 do
17             i ← j
18             j ← arg max_{j∈N\{u}} {x_{ij}}
19             Append j to u
20             k ← k + 1
21             if Σ_{i=1}^{k-1} Σ_{j∈L_i(u)} x_{u_i,u_j} ≤ k − 2 then
22                 Break
23             B_{jγ} calculated as in (6)
24             if B_{jΓ^t} > l_j then
25                 feasible_path ← false
26                 λ ← B_{jΓ^t} − l_j
27                 for r ← k − 2 to 1 do
28                     θ ← B_{u_r,Γ^t} − e_{u_r}
29                     if θ < λ then
30                         Add RIPEC(u_r, ..., u_k) to the pool of inequalities
31                         E ← E ∪ {u_1, ..., u_r}
32                         Break
33                 Break
34         if feasible_path then
35             E ← E ∪ {u_1, ..., u_k}
36     return pool of inequalities
```

6. 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, while the data is available in the supplementary material at https://github.com/abreualexp/robust-pdptw.

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).

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 et al., 2007). During arc elimination, both demand and travel time deviations are considered. 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 $e_i + \bar{t}_{ij} + \hat{t}_{ij} > l_j$. The pool of inequalities from Cordeau (2006) is implemented with additional steps to verify if some routes are feasible in the presence of uncertainty.

Furthermore, we provide an initial robust feasible solution to initialise the methods. This initial solution is generated using a simple 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.

6.1. Instances description

In our experiments, we consider the benchmark instances described in Furtado et al. (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 the budget 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 $\bar{o}_i := Q - \lceil 0.3 \times \bar{o}_i \rceil$. 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.

6.2. 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.

Table 3 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 (*Memory 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 6 for detailed results in terms of Γ^q). The `RPDPTW-2` reached the memory limit in only four instances (one with travel

Uncertainty		`RPDPTW-1`			`RPDPTW-2`		
Demand	Travel time	Opt. (%)	Time lim. (%)	Memory lim. (%)	Opt. (%)	Time lim. (%)	Memory 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 3: Summary of instance solution status for different uncertainty configuration.

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 4, 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.

n	RPDPTW-1				RPDPTW-2			
	Opt. (%)	Time (s)	Gap (%)	Ratio (%)	Opt. (%)	Time (s)	Gap (%)	Ratio (%)
5	100.0	0.0		0.0	100.0	0.0		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 4: Average results of each method among the different instance sizes.

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 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.

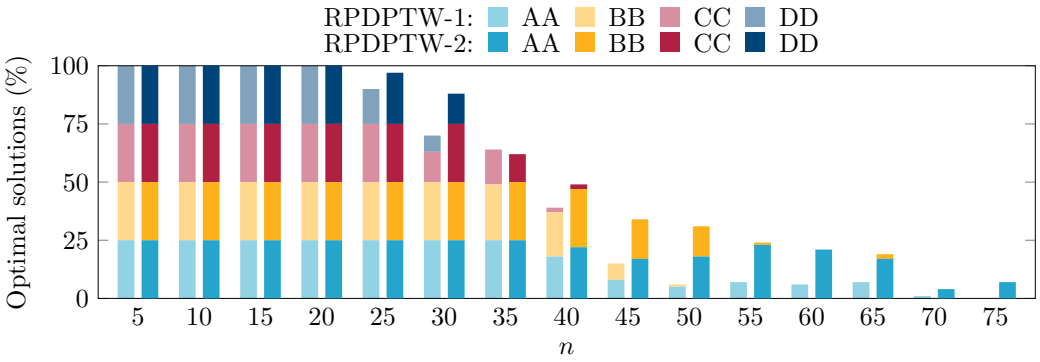
In Table 5, we analyse how each formulation performs under different uncertainty settings across each instance class (AA, BB, CC, and DD). The table presents the same metrics as those shown in Table 4. RPDPTW-2 consistently finds more optimal solutions and requires less computation time than RPDPTW-1. 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. 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

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

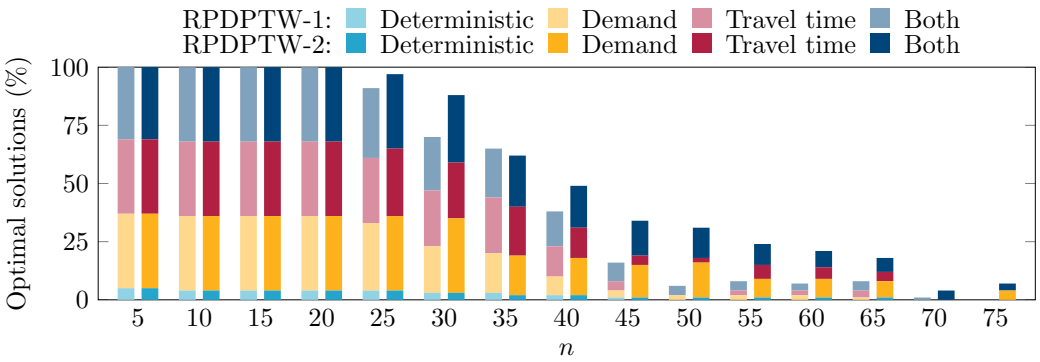
Table 5: Average results for instance classes and uncertainty configurations.

seconds for RPDPTW-2.

Figures 1a and 1b 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 1a, results are grouped into classes, with the stacked bars (from bottom to top) corresponding to classes AA, BB, CC, and DD. In Figure 1b, 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 1b, 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.



(a) Optimal solutions for each instance class.



(b) Optimal solutions for each uncertainty configuration.

Figure 1: Optimal solutions (in percentage) with each method for different instance sizes and characteristics.

In Table 6, 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.

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 6: Average results of each method for the different levels of uncertainty.

6.3. 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 $Obj_{Deterministic}$ 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 3. In the case of deterministic instances (12 instances), where no deviation levels or uncertainty budgets are considered, we adapted the simulation procedure. Specifically, Algorithm 3 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 $\Gamma^t > 0) \times (0.1, 0.2, 0.3)$.

Algorithm 3: Risk Computation

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

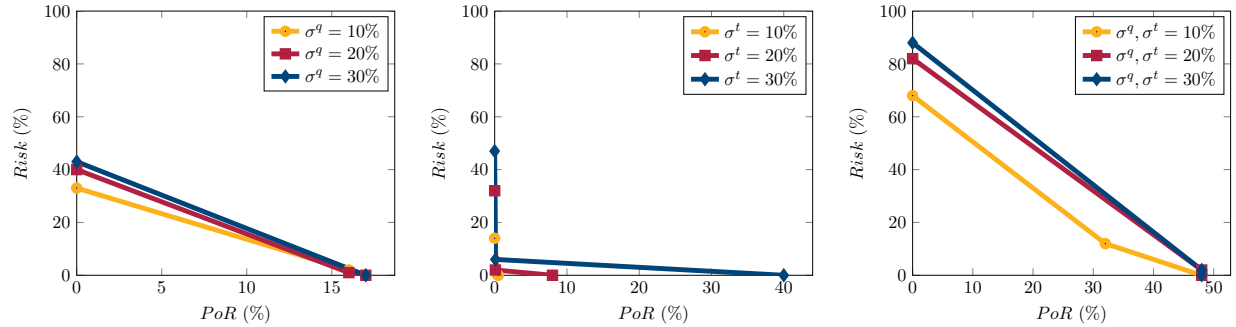
Table 7 presents the average results of *PoR* and *Risk* for each combination of uncertainty budget and deviation level. Moreover, the last two columns display the average results across all deviation values. A *PoR* or *Risk* value of zero is represented by a "-" in the table. The first set of experiments (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 7: PoR and Risk (in percentage) for different deviation and uncertainty levels.

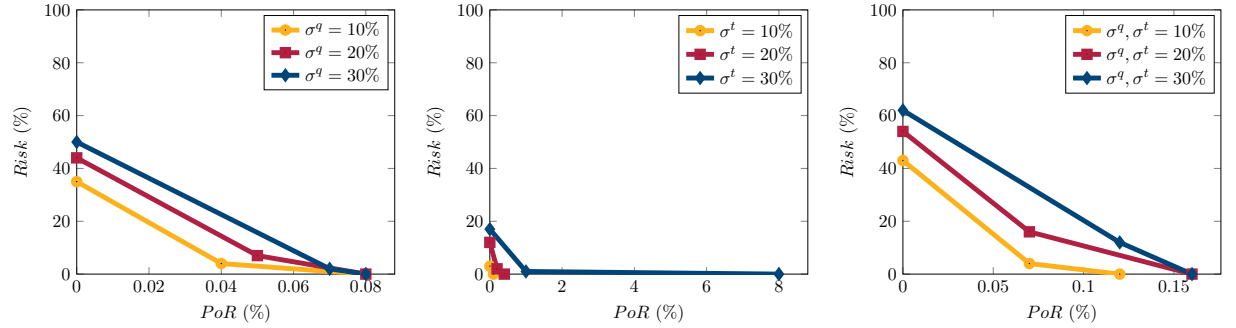
Figure 2 shows the relation between *PoR* and *Risk* grouped in instances with varying values of capacity and time windows length. We can now observe how different capacity and time windows length influence the cost and the feasibility of robust solutions. In particular, we show the results for instances with: Tight capacity (classes AA and CC) and uncertain demands (Figure 2a); Tight time windows (classes AA and BB) and uncertainty travel times (Figure 2b); Tight capacity and time windows (class AA) and both uncertainties (Figure 2c); Loose capacity (classes BB and DD) and uncertain demands (Figure 2d); Loose time windows (classes CC and DD) and uncertain travel times (Figure 2e); and, finally, loose capacity and time windows (class DD) and both uncertainties (Figure 2f). 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) Tight time windows under travel time uncertainty

(c) Tight capacity and time windows under both uncertainties



(d) Loose capacity under demand uncertainty

(e) Loose time windows under travel time uncertainty

(f) Loose capacity and time windows under both uncertainties

Figure 2: Trade-off between PoR and Risk for different types of instances and uncertainty configuration

7. Conclusion

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. We proposed a closed-form equation to compute the worst-case vehicle load in the presence of non-monotonic behaviour, while respecting the pairing property. 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.

Future research directions include exploring the applicability of heuristic and metaheuristic methods for solving large-scale instances of the RPDPTW. Furthermore, the integration of other uncertainty sets, such as the knapsack uncertainty set, could offer valuable insights into balancing solution robustness and cost efficiency within this complex logistics structure. Finally, the proposed approaches can be extended to other related variants of the pickup and delivery problem.

Acknowledgments

This study was funded, in part, by the São Paulo Research Foundation (FAPESP), Brazil (grant numbers 2023/08678-8, 2022/10993-6, 2022/05803-3, 2013/07375-0); the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil (Finance Code 001); the National Council for Scientific and Technological Development (CNPq), Brazil (grant numbers 314079/2023-8, 405702/2021-3); and the Paraíba State Research Foundation (FAPESQ), Brazil (grant number 041/2023). The work reported in this paper was also undertaken as part of the Made Smarter Innovation: Centre for People-Led Digitalisation, at the University of Bath, University of Nottingham, and Loughborough University.

References

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., and Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers & Operations Research*, 40:856–866.

Agra, A., Christiansen, M., Figueiredo, R., Magnus Hvattum, L., Poss, M., and Requejo, C. (2012). Layered formulation for the robust vehicle routing problem with time windows. In *Combinatorial Optimization: Second International Symposium, ISCO 2012, Athens, Greece, April 19-21, 2012, Revised Selected Papers 2*, pages 249–260. Springer.

Al Chami, Z., Bechara, B., Manier, H., and Ange-Manier, M. (2018). A robust pickup and delivery problem with uncertain travel time. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2018-Novem:940–946.

Al Chami, Z., Bechara, B., Manier, H., Manier, M.-A., and Sleiman, M. (2022). A grasp-alns combination for robust pickup and delivery problem. *International Journal of Production Research*, 60:3809–3828.

Allahyari, S., Yaghoubi, S., and Woensel, T. V. (2021). The secure time-dependent vehicle routing problem with uncertain demands. *Computers & Operations Research*, 131:105253.

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, 36(2):69–79.

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

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

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15:1–31.

Berbeglia, G., Cordeau, J. F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202:8–15.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52:35–53.

Campos, R., Coelho, L. C., and Munari, P. (2024). New formulations for the robust vehicle routing problem with time windows under demand and travel time uncertainty. *OR Spectrum*, pages 1–43.

Cordeau, J.-F. (2006). A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586.

Cordeau, J.-F., Laporte, G., and Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 327–357. Springer US.

De La Vega, J., Munari, P., and Morabito, R. (2019). Robust optimization for the vehicle routing problem with multiple deliverymen. *Central European Journal of Operations Research*, 27:905–936.

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

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22.

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

Gendreau, M., Jabali, O., and Rei, W. (2016). 50th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, 50:1163–1173.

Golsefidi, A. H. and Jokar, M. R. A. (2020). A robust optimization approach for the production-inventory-routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, 143:106388.

Gounaris, C. E., Wiesemann, W., and Floudas, C. A. (2013). The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, 61:677–693.

Guo, F., Wang, Z., Huang, Z., and Ma, X. (2024). Robust optimization of microhub network and mixed service strategy for a multidepot location-routing problem. *Computers & Industrial Engineering*, 190:110070.

Hansuwa, S., Kumar, M. R. V., and Chandrasekharan, R. (2022). 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*, 47:1–23.

Huang, X., Chen, H., Liu, L., and Chen, J. (2020). Robust optimization model of feeder lines routing based on the hub port. *Transportation Journal*, 59:279–303.

Kallehauge, B., Boland, N., and Madsen, O. B. (2007). Path inequalities for the vehicle routing problem with time windows. *Networks*, 49:273–293.

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

Liu, X., Wang, D., Yin, Y., and Cheng, T. C. (2023). Robust optimization for the electric vehicle pickup and delivery problem with time windows and uncertain demands. *Computers & Operations Research*, 151:106119.

Lu, D. and Gzara, F. (2019). The robust vehicle routing problem with time windows: Solution by branch and price and cut. *European Journal of Operational Research*, 275:925–938.

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

Mousavi, S. M. and Vahdani, B. (2017). A robust approach to multiple vehicle location-routing problems with time windows for optimization of cross-docking under uncertainty. *Journal of Intelligent & Fuzzy Systems*, 32:49–62.

Munari, P., Moreno, A., Vega, J. D. L., Alem, D., Gondzio, J., and Morabito, R. (2019). The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, 53:1043–1066.

Ordóñez, F. (2010). Robust vehicle routing. In *Risk and Optimization in an Uncertain World*, pages 153–178. INFORMS.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008a). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58:21–51.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008b). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58:81–117.

Pavlova, E. S. and Shichiyakh, R. A. (2021). A robust mathematical model for vehicle routing problem with simultaneous pickup and delivery and worker allocation. *Industrial Engineering & Management Systems*, 20:201–212.

Pessoa, A. A., Poss, M., Sadykov, R., and Vanderbeck, F. (2021). Branch-cut-and-price for the robust capacitated vehicle routing problem with knapsack uncertainty. *Operations Research*, 69:739–754.

Pourmohammad-Zia, N., Schulte, F., González-Ramírez, R. G., Voß, S., and Negenborn, R. R. (2023). A robust optimization approach for platooning of automated ground vehicles in port hinterland corridors. *Computers & Industrial Engineering*, 177:109046.

Ropke, S. and Cordeau, J.-F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43:267–286.

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

Santos, M. J., Curcio, E., Mulati, M. H., Amorim, P., and Miyazawa, F. K. (2020). A robust optimization approach for the vehicle routing problem with selective backhauls. *Transportation Research Part E: Logistics and Transportation Review*, 136:101888.

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

Tajik, N., Tavakkoli-Moghaddam, R., Vahdani, B., and Mousavi, S. M. (2014). A robust optimization approach for pollution routing problem with pickup and delivery under uncertainty. *Journal of Manufacturing Systems*, 33:277–286.

Vakili, R., Shirazi, M. A., and Gitinavard, H. (2021). Multi-echelon green open-location-routing problem: A robust-based stochastic optimization approach. *Scientia Iranica*, 28:985–1000.

Wang, A., Subramanyam, A., and Gounaris, C. E. (2022). Robust vehicle routing under uncertainty via branch-price-and-cut. *Optimization and Engineering*, 23:1895–1948.

Zang, X., Zhu, Y., Zhong, Y., and Chu, T. (2022). Citespace-based bibliometric review of pickup and delivery problem from 1995 to 2021. *Applied Sciences*, 12:4607.

Çağrı Koç and Laporte, G. (2018). Vehicle routing with backhauls: Review and research perspectives. *Computers & Operations Research*, 91:79–91.

Çağrı Koç, Laporte, G., and İlknur Tükenmez (2020). A review of vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, 122:104987.