The robust vehicle routing problem with synchronisation: models and branch-and-cut algorithms

Ricardo Soares*, [a, b], Sophie N. Parragh[c], Alexandra Marques[a], Pedro Amorim[a, b]

[a]INESC TEC – Institute for Systems and Computer Engineering, Technology and Science, Porto, Portugal
[b]FEUP – Faculty of Engineering, University of Porto, Porto, Portugal
[c]Institute of Production and Logistics Management / JKU Business School, Johannes Kepler University Linz, Linz, Austria

Networks
Accepted manuscript, 13th May 2025
https://doi.org/10.1002/net.22287

Abstract

The Vehicle Routing Problem with Synchronisation (VRPSync) aims to minimise the total routing costs while considering synchronisation requirements that must be fulfilled between tasks of different routes. These synchronisation requirements are especially relevant when it is necessary to have tasks being performed by vehicles within given temporal offsets, a frequent requirement in applications where multiple vehicles, crews, materials or other resources are involved in certain operations. Although several works in the literature have addressed this problem, mainly the deterministic version has been tackled so far. This paper presents a robust optimisation approach for the VRPSync, taking into consideration the uncertainty in vehicle travel times between customers. This work builds on existing approaches in the literature to develop mathematical models for the Robust VRPSync, as well as a branch-and-cut algorithm to solve more difficult problem instances. A set of computational experiments is also devised and presented to obtain insights regarding key performance parameters of the mathematical models and the solution algorithm. The results suggest that solution strategies where certain standard problem constraints are only introduced if a candidate solution violates any of those constraints provide more consistent improvements than approaches that rely on tailor-made cutting planes, added through separation routines. Furthermore, the analysis of the Price of Robustness indicators shows that the adoption of robust solutions can have a significant increase in the total costs, however, this increase quickly plateaus as budgets of uncertainty increase.

Keywords: vehicle routing, synchronisation, robust optimisation, travel time uncertainty, mathematical programming, branch-and-cut

*Corresponding author.
E-mail addresses: ricardo.f.soares@inesctec.pt (R. Soares)

1 Introduction

The Vehicle Routing Problem with Synchronisation (VRPSync) is an extension of the standard Vehicle Routing Problem (VRP) where routes may be "interrelated" in terms of their feasibility. This definition covers problems where different vehicles need to visit customers within given time offsets – as it happens with certain logistics operations requiring more than one vehicle for them to be completed –, or when different vehicles need to travel simultaneously in certain portions of a route. This problem variant can be found in specific applications where vehicles, specialised staff or resources need to have their unproductive times minimised, such as home health care routing (e.g., Bredström & Rönnqvist, 2008; Mankowska, Meisel, & Bierwirth, 2013), public utilities maintenance (e.g., Goel & Meisel, 2013; Hà et al., 2020), routing with autonomous vehicles (e.g., Coindreau, Gallay, & Zufferey, 2021; Li, Wang, Chen, & Bai, 2020) or truck and trailer routing (e.g., Chao, 2002; Meisel & Kopfer, 2012). Due to the specificities of the VRPSync compared to traditional VRPs, this problem variant poses the additional challenge that each route of a given solution cannot be evaluated independently.

In the literature, stochastic programming and robust optimisation constitute the two main approaches for dealing with uncertainty in VRPs (Averbakh, 2001). In stochastic programming, knowing the probability distributions of the uncertain parameters or at least being able to estimate them is usually required. Its goal consists of optimising a solution's expected value while retaining solution feasibility for a variety of alternative scenarios (Birge & Louveaux, 2011). On the other hand, robust optimisation does not require that probability distributions be known. This approach assumes that the values taken by the uncertain parameters are contained within an uncertainty set and the optimal solution must be feasible for all possible realisations of the uncertainty sets (e.g., Ordóñez, 2010). In practice, these assumptions usually result in optimisation models where the worst-case scenarios of the uncertain parameters (i.e., the limits of the uncertainty sets) must be considered in the problem, and the optimal solution must be feasible for all worst-case values that the uncertain parameters may take.

One of the main criticisms of robust optimisation is its excessive conservatism. To control the degree of conservatism, Bertsimas and Sim (2004) proposed an approach where the number of worst-case occurrences that a robust-feasible solution must be able to accommodate is limited to a given value, Γ. The parameter Γ is typically called the budget of uncertainty.

In this paper, we address the VRPSync with travel time uncertainty and we present a robust optimisation approach. To the best of our knowledge, robust optimisation has not yet been applied to the VRPSync. This paper leverages existing knowledge from the literature concerning robust optimisation and vehicle routing to obtain mathematical formulations for the Robust VRPSync. For solving larger instances, a branch-and-cut algorithm is implemented that considers the specificities of the problem being solved.

The remainder of this paper is as follows. Section 2 provides an overview of previous work on the topics of routing with synchronisation under uncertainty and robust vehicle routing. Section 3 states and describes the Robust VRPSync subject to uncertainty in travel times. Section 4 presents the solution method developed to solve this problem. Section 5 designs computational experiments and presents the obtained results. Finally, Section 6 states the conclusions and provides insights into the developed work, as well as future research.

2 Related work

This section provides an overview of existing work that relates to several aspects of the problem tackled in this paper. Initially, it frames the VRPSync considering existing literature and concepts. Aftwerwards, a summary on existing vehicle routing approaches that use robust optimisation is provided.

2.1 Routing with synchronisation

The concept of synchronisation in vehicle routing was coined by Drexl (2012), which provided a survey and classification schema for synchronisation. The survey was recently updated in a literature review by Soares, Marques, Amorim, and Parragh (2024), which defines synchronisation as a problem requirement that makes two or more routes interdependent, making their feasibility mutually dependent and in need to be assessed jointly.

Routing problems with synchronisation usually require the adoption of additional modelling concepts (Drexl, 2012). Therefore, concepts such as locations, tasks, arcs and operations must frequently be distinguished. Soares et al. (2024) emphasise that, while locations typically refer to real-world sites, tasks correspond to nodes in the transportation network, which can be visited by a vehicle. Depending on the problem assumptions, a location may contain various tasks, since it may need to be visited more than once. Furthermore, arcs and operations must be distinguished. While arcs correspond to pairs of tasks belonging to the problem's transportation network, operations correspond to associations of tasks that are interrelated. In synchronisation problems, operations typically represent groups of tasks that need to be synchronised (e.g., multiple vehicles synchronising their arrival times to a given location).

2.1.1 Synchronisation of schedules

Soares et al. (2024) consider two major types of synchronisation: movement synchronisation and operation synchronisation. Movement synchronisation applies when a given vehicle is limited on the arcs it can traverse, depending on the arcs traversed by another vehicle. Representative examples of this synchronisation type include truck-and-trailer routing (e.g., Chao, 2002) or whenever a vehicle cannot move autonomously without the assistance of another moving vehicle, such as drayage transports (e.g., Meisel & Kopfer, 2012). Operation synchronisation occurs when the arrival time of a vehicle to a certain task depends on the arrival time of another vehicle to a certain task. Within the two major types of synchronisation, this paper only considers operation synchronisation, since it is the synchronisation aspect that is inherently impacted by travel time uncertainty.

Considering the predominant applications of routing problems with synchronisation proposed by Soares et al. (2024), this problem can be characterised as a routing problem with synchronisation of schedules, a type of scheduling-centric routing problem with a significant prevalence of operation synchronisation constraints. Home health care routing (e.g., Bredström & Rönnqvist, 2008; Mankowska et al., 2013) is a common application of this problem class. In home health care routing, staff members have different qualifications to perform certain services at customer locations. Some customers require multiple caregivers to perform an operation, whose tasks may need to be synchronised to be performed simultaneously or sequentially. Other routing problems with synchronisation of schedules can be found in the literature, such as synchronising the maintenance tasks on both ends of utility lines to minimise downtime (e.g., Goel & Meisel, 2013) or synchronising the charging tasks of different electric vehicles, taking into account the limited availability of charging stations (e.g., Froger, Jabali, Mendoza, & Laporte, 2022).

2.1.2 Synchronisation under uncertainty

The study of the VRPSync has largely been centred on deterministic scenarios, with only a minimal number of studies addressing its uncertainty aspects. Notably, Furian, O'Sullivan, Walker, and Vössner (2018) have explored a home health care routing problem that involves operation synchronisation, introducing an optimisation technique within a discrete-event simulation framework to manage stochastic elements. Similarly, Hashemi Doulabi, Pesant, and Rousseau (2020) have examined a VRP variant that includes synchronised visits alongside stochastic travel and service durations. Another contribution by Shi, Zhou, Ye, and Zhao (2020) deals with a routing problem with synchronised visits focusing on minimising greenhouse gas emissions and employing robust optimisation to tackle uncertainties in travel and service times. Anderluh, Larsen, Hemmelmayr, and Nolz (2019) have applied a Monte Carlo simulation combined with optimisation strategies to a 2-echelon VRP that synchronises vans and bicycles at satellite locations under uncertain travel conditions. Mourad, Puchinger, and Van Woensel (2020) have discussed a pickup and delivery problem involving autonomous robots that could leverage public transportation to expand their operational area. The autonomous robots must synchronise with the arrival of the public transportation for them to be transported by it. The passengers are given priority over the autonomous robots, so the usage of the robots is only possible if it does not compromise passenger demand, which is deemed uncertain.

The limited discussion of uncertainty in the context of routing with synchronisation demonstrates the underdevelopment of this research area, an observation that has already been acknowledged by existing work (e.g., Parragh & Doerner, 2018; Soares et al., 2024).

2.2 Robust vehicle routing

Developments in robust optimisation have been motivated by the fact that, in many problem settings, there is no clear knowledge of the distribution of the uncertain parameters of the problem. Therefore, it attempts to mitigate one of the vulnerabilities of stochastic optimisation, which is the need to know (or be able to estimate) the probability distribution functions of the uncertain parameters. Soyster (1973) laid the groundwork for this research field by proposing a robust optimisation approach where it is assumed that a solution should be robust-feasible when it is able to remain feasible when all uncertain parameters take their worst-case values. Later, Bertsimas and Sim (2004) proposed an alternative approach that limits over-conservativeness by considering a fixed budget of uncertainty. The work presented in this paper is based on this latter approach.

Robust vehicle routing has been a target of study for several years. Sungur, Ordóñez, and Dessouky (2008) present one of the first works applying robust optimisation approaches in this field, which considered uncertainty in customer demand. The work studied multiple types of uncertainty sets and assumed that robust feasibility would require a solution to be feasible for all possible realisations of customer demand in the uncertainty set.

Later works attempt to mitigate the over-conservativeness of robust optimisation by bounding uncertainty. Ordóñez (2010) outlines different robust routing problems taking into account the robust optimisation approach of Bertsimas and Sim (2004), which studied varying sources of uncertainty, such as time, demand, costs or customers.

The literature suggests that customer demand and travel times are the most common sources of uncertainty and are often considered together. Lee, Lee, and Park (2012) consider a problem with uncertainty in both customer demand and travel times, considering two types of uncertainty sets and resorting to a branch-and-price method to solve the problem. Hu, Lu, Liu, and Zhang (2018) also tackle a VRP with Time Windows (VRPTW) with uncertainty in demand and travel times

and propose an adaptive heuristic to solve the problem. Hoogeboom, Adulyasak, Dullaert, and Jaillet (2021) include uncertainty in service and travel times in a routing problem focused on time window assignments. Munari et al. (2019) also propose a compact formulation for the VRPTW with uncertainty in demand and travel times. The formulation provides recursive definitions for vehicle loads and customer arrival times, which are incorporated into the problem constraints afterward. The work envisaged in this paper builds significantly from this work.

Other works only consider a single parameter subject to uncertainty. Agra et al. (2012, 2013) propose formulations for an uncapacitated VRPTW and consider bounded uncertainty in the travel times defined within an uncertainty polytope. Customer demand is also very commonly considered (e.g., Carlsson & Delage, 2013; Gounaris, Wiesemann, & Floudas, 2013; Lu & Gzara, 2019). Other alternative sources of uncertainty, such as revenues (e.g., Santos, Curcio, Mulati, Amorim, & Miyazawa, 2020), are seldom found.

3 Problem description and formulation

3.1 The vehicle routing problem with synchronisation

This section outlines the mathematical model for the VRPSync and the theoretical assumptions behind it, adopting the concepts and nomenclature outlined in the previous section.

The VRPSync generalises the VRPTW. The objective is to determine the optimal routes for a set of K vehicles, initially located at a depot 0, to perform a set of n tasks in a set of m geographically dispersed locations, all while respecting the capacity limits of each vehicle, Q, fulfilling the demand of each task i, $q_i \geq 0$.

When performing a given task, the arrival time of the vehicle must be bounded within a closed interval $[a_i, b_i]$, thus respecting the time windows established for that task. To correctly compute task arrival times, travel times between tasks must be considered (parameter d_{ij}), as well as the service time required to perform each task at its corresponding location (parameter s_i).

In addition to these requirements, the optimal routes must respect given operation synchronisation constraints, which establish time offset requirements between the arrival times of tasks being performed in different routes. For these synchronised tasks, time offsets between the arrival times of two tasks of an operation (o, p) must be considered, which are represented by parameters λ_{op} and μ_{op}. Dohn, Rasmussen, and Larsen (2011) proposed different combinations of these synchronisation constraints, which can be decomposed into two types: (i) lower-bounding constraints (LB), which set the minimum time offset λ_{op} between o and p; and (ii) upper-bounding constraints (UB), which set the maximum time offset μ_{op} between o and p.

In this paper, no other synchronisation aspect beyond operation synchronisation is considered.

Table 1 presents the mathematical notation in terms of sets and parameters for the problems presented in this paper.

3.2 The robust vehicle routing problem with synchronisation

The Robust VRPSync further generalises the VRPSync by considering that certain problem parameters are uncertain and whose values may take values within a previously defined uncertainty set.

In this paper, we consider that the uncertain parameters are the vehicle travel times between tasks. To that effect, we base ourselves on the notation and approach adopted by Munari et al. (2019), who tackled the Robust VRPTW. As such, the travel times between tasks are subject to the polyhedral uncertainty set defined in Equation (1):

Table 1: List of sets and parameters

Sets		
$\mathcal{L}$	Set of locations	
$\mathcal{N}$	Set of tasks	$\mathcal{N} = \{1, 2, ..., n\}$
$\mathcal{N}_0$	Set of tasks with depot tasks	$\mathcal{N}_0 = \mathcal{N} \cup \{0, n+1\}$
$\mathcal{A}$	Set of arcs (i, j)	$\mathcal{A} \subset \mathcal{N}_0 \times \mathcal{N}_0$
$\mathcal{R}$	Set of all operations (o, p)	$\mathcal{R} \subset \mathcal{N} \times \mathcal{N}$
$\mathcal{R}^\lambda$	Set of operations (o, p) subject to lower-bounding synchronisation constraints	$\mathcal{R}^\lambda \subseteq \mathcal{R}$
$\mathcal{R}^\mu$	Set of operations (o, p) subject to upper-bounding synchronisation constraints	$\mathcal{R}^\mu \subseteq \mathcal{R}$
Parameters		
General parameters:		
c_{ij}	Cost for traversing arc $(i, j) \in \mathcal{A}$	
d_{ij}, d'_{ij}	Travel time, distance from the location of task i to the location of task j	
$\hat{d}_{ij}$	Maximum positive travel time deviation when travelling from the location of task i to the location of task j	
Q	Capacity of each vehicle	
T	Planning horizon	
s_i	Service time of task $i \in \mathcal{N}_0 \setminus \{n+1\}$	
Parameters related with tasks:		
a_i, b_i	Earliest, latest possible time to begin performing task $i \in \mathcal{N}_0$	
e_i	takes value 1, if task i is mandatory; 0, otherwise	
q_i	Demand to be satisfied for task $i \in \mathcal{N}_0$	
Parameters related with operations:		
e_{op}	takes value 1, if operation $(o, p) \in \mathcal{R}$ is mandatory; 0, otherwise	
λ_{op}	Min. time offset between the arrival times of synchronised operation $(o, p) \in \mathcal{R}^\lambda$	
μ_{op}	Max. time offset between the arrival times of synchronised operation $(o, p) \in \mathcal{R}^\mu$	

$$\mathscr{U} = \left\{ \tilde{d} \in \mathbb{R}_{+}^{|\mathcal{A}|} : \tilde{d}_{ij} = d_{ij} + \hat{d}_{ij}\xi_{ij}, \sum_{(i,j)\in\mathcal{A}} \xi_{ij} \leq \Gamma, 0 \leq \xi_{ij} \leq 1, \forall (i,j) \in \mathcal{A} \right\} \tag{1}$$

with d_{ij} representing the nominal travel time between tasks i and j, $\tilde{d}_{ij}$ representing the random variables for the values of travel times between i and j and $\hat{d}_{ij}$ representing the positive travel time deviations from their nominal values, d_{ij}.

Under this theoretical setting, it is necessary to determine how arrival times can be defined and modelled when operation synchronisation is at play. Considering the nomenclature already presented in Table 1, $\mathcal{R}^{\lambda}$ and $\mathcal{R}^{\mu}$ represent the sets of operations that are subject to lower-bounded and upper-bounded synchronisation, respectively, and $\mathcal{R} = \mathcal{R}^{\lambda} \cup \mathcal{R}^{\mu}$ aggregates the operations from both sets.

Let us assume there are no "chained" synchronisation constraints, meaning that a task cannot be subject to operation synchronisation with more than one other task. Let k be a route that performs a sequence of $r+1$ tasks $(v_0, v_1, ..., v_r)$, where $v_0 = 0$ and $v_r = n+1$. In this context, let $t^{\gamma}_{v_j}$ be the earliest time the service can start at node v_j when up to $\gamma \leq \Gamma$ travel times reach their worst-case values. Also, let us postulate that the maximum value of an empty set equals zero ($\max\{\emptyset\} = 0$). Building on the recursive definition of Munari et al. (2019), the values of $t^{\gamma}_{v_j}$ can be calculated as presented in Equation (2).

$$t^{\gamma}_{v_j} = \begin{cases} a_{v_0}, & \text{if } j = 0; \\ \max\Big\{ a_{v_j}, t^{\gamma}_{v_{j-1}} + s_{v_{j-1}} + d_{v_{j-1}v_j}, & \\ \quad \max\limits_{o:(o,v_j)\in\mathcal{R}^{\lambda}} \left\{ t^{\Gamma}_{o} + \lambda_{ov_j} \right\}, \max\limits_{p:(v_j,p)\in\mathcal{R}^{\mu}} \left\{ t^{\Gamma}_{p} - \mu_{v_jp} \right\} \Big\}, & \text{if } \gamma = 0; \\ \max\Big\{ a_{v_j}, t^{\gamma}_{v_{j-1}} + s_{v_{j-1}} + d_{v_{j-1}v_j}, t^{\gamma-1}_{v_{j-1}} + s_{v_{j-1}} + d_{v_{j-1}v_j} + \hat{d}_{v_{j-1}v_j}, & \\ \quad \max\limits_{o:(o,v_j)\in\mathcal{R}^{\lambda}} \left\{ t^{\Gamma-\gamma}_{o} + \lambda_{ov_j} \right\}, \max\limits_{p:(v_j,p)\in\mathcal{R}^{\mu}} \left\{ t^{\Gamma-\gamma}_{p} - \mu_{v_jp} \right\} \Big\}, & \text{otherwise.} \end{cases} \tag{2}$$

A route will be considered robust-feasible if $t^{\gamma}_{v_j} \leq b_{v_j}, \forall \gamma = 0, 1, ..., \Gamma, j = 1, ..., r$.

The rationale behind this calculation is as follows. The presence of operation synchronisation impacts the arrival times of the tasks that constitute an operation, which ultimately impacts the values of $t^{\gamma}_{v_j}$.

Considering an operation (o, p) involved in synchronisation and their corresponding arrival times t^{γ}_{o} and t^{γ}_{p}, it is intuitive to state that the earliest arrival times of tasks o and p are mutually dependent. This leads to the conclusion that, for these tasks, it is necessary to consider the number of worst-case deviations occurring on both routes, since each route has a distinct budget of uncertainty. The proposed recursive definition considers that the synchronisation of o with p must be such that both tasks are not subjected to more than Γ worst-case deviations in both routes. In other words, $\gamma + \gamma' = \Gamma$, leading to the conclusion that t^{γ}_{o} must be computed taking into consideration the value of $t^{\Gamma-\gamma}_{p}$.

For a task not subject to any operation synchronisation constraints, the above calculation simplifies to precisely the one provided by Munari et al. (2019).

3.3 Mathematical formulations

In Appendix A the reader can find a mathematical formulation for the deterministic version of the VRPSync, which serves as a basis for the robust models presented ahead.

The Robust VRPSync is outlined through three vehicle flow formulations. The devised formulations build on the modelling framework of Soares et al. (2024), from which several simplifications were considered, taking into account the focus of this paper. Three alternative formulations are presented, namely (i) one formulation where the routing variables contain three indices, which correspond to the tasks of the arc being traversed and the vehicle traversing it (variables x_{ij}^k); and (ii) two formulations where the routing variables only contain two indices corresponding to the arcs being traversed by a vehicle (variables x_{ij}).

The 3-index formulation is general and therefore applies to all problem settings. However, one of the 2-index formulations is only applicable under certain conditions. In fact, due to the inherent inter-route dependencies that the VRPSync entails, routing variables without an index representing the vehicle/route can result in additional model complexity when accounting for interdependencies between routes. Nevertheless, these 2-index models were devised taking into account existing results from the literature that robust optimisation models with 3-index routing formulations are typically cumbersome (e.g., Agra et al., 2012; Lee et al., 2012). The general 3-index formulation corresponds to model [RVRPSync3] and is presented in Appendix B to provide a more streamlined experience to the reader.

The 2-index formulations are henceforth presented. The problem relies on a transportation network, which is defined through $\mathcal{G} = (\mathcal{N}_0, \mathcal{A})$, with $\mathcal{A} = \{(i,j) \in \mathcal{N}_0 \times \mathcal{N}_0 : i \neq j, i \neq n+1, j \neq 0\}$. We consider the following decision variables:

$$x_{ij} \quad \begin{cases} 1 & \text{if arc } (i,j) \in \mathcal{A} \text{ is traversed} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$y_{op} \quad \begin{cases} 1 & \text{if operation } (o,p) \in \mathcal{R} \text{ is performed} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$u_{ij} \quad \text{Load of vehicle when traversing arc } (i,j) \in \mathcal{A} \tag{5}$$

$$t_i^\gamma \quad \text{Earliest arrival time at task } i \in \mathcal{N}_0 \text{ when } \gamma \leq \Gamma \text{ travel times have so far reached their worst-case values} \tag{6}$$

Note: Variables y_{op} may be relaxed to continuous variables.

Model [RVRPSync1] This formulation consists of a compact, simplified formulation, which is applicable if it is possible to guarantee, through the problem's parameters, that all synchronised operations cannot possibly be performed by the same vehicle. Table 2 outlines the assumptions under which this model is applicable.

Under these conditions, the Robust VRPSync can be formulated as the following mixed-integer linear program:

$$\text{[RVRPSync1]} \qquad \min \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \tag{7}$$

subject to

$$e_j \leq \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{8}$$

Table 2: Adopted assumptions for model [RVRPSync1]

Triangle Inequality Assumption (d_{op})	Γ	Model [RVRPSync1] is applicable if	
Not valid	$\Gamma = 0$	$s_o > \min\{\lambda_{op}, b_p - a_o - d_{op}\}$	$\forall (o,p) \in \mathcal{R}^{\lambda}$
		$s_o > \min\{\mu_{op}, b_p - a_o - d_{op}\}$	$\forall (o,p) \in \mathcal{R}^{\mu}$
	$\Gamma > 0$	$s_o > \min\left\{\lambda_{op}, b_p - a_o - d_{op} - \hat{d}_{op}\right\}$	$\forall (o,p) \in \mathcal{R}^{\lambda}$
		$s_o > \min\left\{\mu_{op}, b_p - a_o - d_{op} - \hat{d}_{op}\right\}$	$\forall (o,p) \in \mathcal{R}^{\mu}$
Valid	$\Gamma = 0$	$s_o > \min\{\lambda_{op} - d_{op}, b_p - a_o - d_{op}\}$	$\forall (o,p) \in \mathcal{R}^{\lambda}$
		$s_o > \min\{\mu_{op} - d_{op}, b_p - a_o - d_{op}\}$	$\forall (o,p) \in \mathcal{R}^{\mu}$
	$\Gamma > 0$	$s_o > \min\left\{\lambda_{op} - d_{op} - \hat{d}_{ij}, b_p - a_o - d_{op} - \hat{d}_{op}\right\}$	$\forall (o,p) \in \mathcal{R}^{\lambda}$
		$s_o > \min\left\{\mu_{op} - d_{op} - \hat{d}_{op}, b_p - a_o - d_{op} - \hat{d}_{op}\right\}$	$\forall (o,p) \in \mathcal{R}^{\mu}$

Note: The triangle inequality assumption is valid for parameter d_{op} iff $d_{op} \leq d_{ol} + d_{lp}, \forall (o,p),(o,l),(l,p) \in \mathcal{A}$.

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 0 \qquad \forall j \in \mathcal{N} \tag{9}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j} = \sum_{i:(i,n+1)\in\mathcal{A}} x_{i,n+1} \tag{10}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j} \leq K \tag{11}$$

$$u_{ij} \leq Q x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{12}$$

$$\sum_{i:(i,j)\in\mathcal{A}} u_{ij} + q_j \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \leq \sum_{i:(i,j)\in\mathcal{A}} u_{ji} \qquad \forall j \in \mathcal{N}_0 \setminus \{0\} \tag{13}$$

$$t_i^{\gamma} + (s_i + d_{ij})\, x_{ij} \leq t_j^{\gamma} + T(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A}, \gamma = 0, ..., \Gamma \tag{14}$$

$$t_i^{\gamma-1} + \left(s_i + d_{ij} + \hat{d}_{ij}\right) x_{ij} \leq t_j^{\gamma} + T(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A}, \gamma = 1, ..., \Gamma \tag{15}$$

$$\sum_{i:(i,j)\in\mathcal{A}} a_j x_{ij} \leq t_j^{\gamma} \leq \sum_{i:(i,j)\in\mathcal{A}} b_j x_{ij} \qquad \forall j \in \mathcal{N}, \gamma = 0, ..., \Gamma \tag{16}$$

$$\sum_{l:(l,o)\in\mathcal{A}} x_{lo} + \sum_{l:(l,p)\in\mathcal{A}} x_{lp} - 1 \leq y_{op} \qquad \forall (o,p) \in \mathcal{R} \tag{17}$$

$$y_{op} \leq \sum_{l:(l,o)\in\mathcal{A}} x_{lo} \qquad \forall (o,p) \in \mathcal{R} \tag{18}$$

$$y_{op} \leq \sum_{l:(l,p)\in\mathcal{A}} x_{lp} \qquad \forall (o,p) \in \mathcal{R} \tag{19}$$

$$e_{op} \leq y_{op} \leq 1 \qquad \forall (o,p) \in \mathcal{R} \tag{20}$$

$$t_o^{\gamma} + \lambda_{op} \leq t_p^{\gamma'} + T\,(1 - y_{op}) \qquad \forall (o,p) \in \mathcal{R}, \gamma, \gamma' \in \mathbb{N}_0 : \gamma + \gamma' = \Gamma \tag{21}$$

$$t_o^{\gamma} + \mu_{op} + T\,(1 - y_{op}) \geq t_p^{\gamma'} \qquad \forall (o,p) \in \mathcal{R}, \gamma, \gamma' \in \mathbb{N}_0 : \gamma + \gamma' = \Gamma \tag{22}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{23}$$

$$y_{op} \in \{0,1\} \qquad \forall (o,p) \in \mathcal{R} \tag{24}$$

$$0 \leq t_i^{\gamma} \leq T \qquad \forall i \in \mathcal{N}_0, \gamma = 0, ..., \Gamma \tag{25}$$

$$0 \leq u_{ij} \leq Q \qquad \forall i \in \mathcal{N}_0 \tag{26}$$

The problem's objective function minimises the total travel costs, as defined in Expression (7).

Constraints (8) ensure that every task is performed at most once, and they force mandatory tasks to be performed; they are *global task constraints*. In this formulation, we assume that no tasks are being performed more than once. Furthermore, it is assumed, for the sake of generality, that not all tasks may necessarily be mandatory. According to Soares et al. (2024), this is a frequent assumption in the context of the VRPSync since it is common to have to acknowledge optional tasks in real-life applications of this problem.

Constraints (9) establish the inflow and outflow conservation: vehicles entering a task node must also exit it. Constraints (10) ensure that every vehicle starts and ends its route at the depot, and constraints (11) bound the number of vehicles used.

Constraints (12) are linking constraints between variables x_{ij} and u_{ij}; they impose that the load of a vehicle when traversing arc (i, j) cannot be different from zero if no vehicle traverses said arc ($x_{ij} = 0 \implies u_{ij} = 0$). Constraints (13) are demand satisfaction constraints: they resort to commodity flow load propagation (e.g., Campos, Coelho, & Munari, 2024) to ensure that the difference between the vehicle load entering the task and the vehicle load leaving the task must account for the demand of the task being performed.

Constraints (14) and (15) establish vehicles earliest arrival times to task nodes. They also serve as sub-tour elimination constraints and are an alternative to the Miller-Tucker-Zemlin constraints. Constraints (16) establish lower and upper bounds to the earliest arrival time of a vehicle to a given task, according to its desired time windows.

Constraints (17)–(19) are linking constraints that set the values of decision variables y_{op} based on the values of variables x_{ij}. Constraints (17) set variable y_{op} to 1 if tasks o and p of operation $(o, p) \in \mathcal{R}$ are being performed. Constraints (18) and (19) set the opposite case: when one of the tasks is not being performed, then the value of y_{op} is forcefully set to zero.

Constraints (20) ensure that operations are performed at most once, and they force mandatory operations to be performed, similarly to what happens in constraints (8).

Constraints (21) correspond to what Soares et al. (2024) consider to be lower-bounding (LB) synchronisation constraints: they ensure that, for a given synchronised operation (o, p), the earliest start time of task p can only be performed λ_{op} time units after the start time of task o. In other words, these constraints establish a lower-bound to the earliest start time of task p based on the earliest start time of task o. Analogously, constraints (22) correspond to upper-bounding (UB) synchronisation constraints: they state that, for a given synchronised operation (o, p), if o is performed before p, then the earliest start time of task p can only be performed up to μ_{op} time units after the start time of o. In other words, these constraints establish an upper-bound to the earliest start time of task p based on the earliest start time of task o.

Constraints (23)–(26) establish the decision variable domains.

Model [RVRPSync2] An alternative formulation with fewer variables is also presented. This formulation is obtained by replacing constraints (12)–(22) with subtour elimination constraints, rounded capacity inequalities, infeasible path elimination constraints and constraints to eliminate infeasible sets of paths. Let $\mathcal{S}$ designate the powerset of $\mathcal{N}$, and let S be an element of $\mathcal{S}$. Under this definition, let $q(S) = \sum_{i \in S} e_i q_i$. Furthermore, denote by $\mathcal{P}$ the set of all infeasible paths with respect to time windows. Considering P as an infeasible path contained in $\mathcal{P}$, we designate $A(P)$ and $N(P)$ as the arc set and node set of path P, respectively. Finally, denote by $\mathcal{T}$ the set of all infeasible sets of paths, i.e., the sets of paths that are infeasible when being used in the same solution. Under this assumption, let T represent an element of $\mathcal{T}$, i.e., an infeasible set of paths. These sets of paths are infeasible with respect to synchronisation constraints.

The problem can therefore be reformulated as follows:

$$\text{[RVRPSync2]} \qquad \min \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{27}$$

subject to:

$$e_j \leq \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{28}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 0 \qquad \forall j \in \mathcal{N} \tag{29}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j} = \sum_{i:(i,n+1)\in\mathcal{A}} x_{i,n+1} \tag{30}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j} \leq K \tag{31}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq \mathcal{S} \tag{32}$$

$$\sum_{i\notin S}\sum_{j\in S} x_{ij} \geq \left\lceil \frac{q(S)}{Q} \right\rceil \qquad \forall S \subseteq \mathcal{S} \tag{33}$$

$$\sum_{(i,j)\in A(P)} x_{ij} \leq |A(P)| - 1 \qquad \forall P \in \mathcal{P} \tag{34}$$

$$\sum_{P\in T}\sum_{(i,j)\in A(P)} x_{ij} \leq \sum_{P\in T} |A(P)| - 1 \qquad \forall T \in \mathcal{T} \tag{35}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{36}$$

The problem's objective function, defined in Expression (27), minimises the total travel costs and is identical to the one in model [RVRPSync1]. Constraints (28)–(31) correspond to constraints (8)–(11) of model [RVRPSync1].

Constraints (32) are subtour elimination constraints. Constraints (33) act as a replacement of capacity constraints (12) and (13), which ensure that vehicle capacity limits are not exceeded.

Constraints (34) act as a replacement of constraints (14)–(16), which establish vehicle arrival times at tasks, as well as time window bounds. This is because set $\mathcal{P}$ is populated with all paths that violate time window bounds in a robust-feasible setting. The determination of infeasible paths is a frequently found approach in the context of vehicle routing (e.g., Ascheuer, Fischetti, & Grötschel, 2000; Enzi, Parragh, & Puchinger, 2021) and their underlying concept is extended in this paper to consider robust-infeasible paths. Under the robustness assumptions of this problem, a robust-infeasible path is a path that is deemed infeasible for any combination of Γ worst-case travel time deviations occurring within that path.

Constraints (35) replace constraints (17)–(22). This is because set $\mathcal{T}$ is populated with all groups of paths that would violate synchronisation constraints if all paths contained in that group occurred in the same solution.

Finally, constraints (36) establish the domain of decision variables x_{ij}.

3.4 Valid inequalities

The proposed models can be strengthened through valid inequalities, which are described in the following.

3.4.1 Subtour elimination constraints

Strengthened subtour elimination constraints – type 1 Taking into account the operations that are subject to synchronisation, it is possible to derive strengthened subtour elimination constraints, building from the standard subtour elimination constraints $x(S) \leq |S| - 1, \forall S \subseteq \mathcal{S}$.

$$\sum_{i,j \in S} x_{ij} \leq |S| - 2 \qquad \forall S \subseteq \mathcal{S}, (o,p) \in \mathcal{R} : o \in S \wedge p \in S \tag{37}$$

$$\sum_{i,j \in S} x_{ij} + \sum_{j \in S} x_{oj} + \sum_{i \in S} x_{io} \leq |S| - 1 \qquad \forall S \subseteq \mathcal{S}, (o,p) \in \mathcal{R} : o \notin S \wedge p \in S \tag{38}$$

$$\sum_{i,j \in S} x_{ij} + \sum_{j \in S} x_{pj} + \sum_{i \in S} x_{ip} \leq |S| - 1 \qquad \forall S \subseteq \mathcal{S}, (o,p) \in \mathcal{R} : o \in S \wedge p \notin S \tag{39}$$

Constraints (37) are valid if set S contains at least two tasks o and p that constitute an operation that is subject to synchronisation. Constraints (38) and (39) are valid if only one of the tasks of a synchronised operation is contained in set S.

Strengthened subtour elimination constraints – type 2 Additional subtour elimination constraints can be introduced in parallel with the ones above. Let $\mathcal{U}_i = \{j \in S : (i,j) \in \mathcal{A}\}$ be auxiliary sets which contain the tasks within S that are directly accessible from $i \in S$. The following inequalities can then be introduced for the problem at hand.

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq |S| - \max_{i \in S} \{e_i \, |\mathcal{U}_i|\} \qquad \forall S \subseteq \mathcal{S} : i \in S \implies \exists! j \in S : (i,j) \in \mathcal{R} \vee (j,i) \in \mathcal{R} \tag{40}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq |S| - \max_{i \in S} \{e_i \, |\mathcal{U}_i|\} \qquad \forall S \subseteq \mathcal{S} : i \in S \implies \exists! j \in S : (i,j) \in \mathcal{R} \vee (j,i) \in \mathcal{R} \tag{41}$$

Constraints (40) and (41) are valid only for subsets of $\mathcal{S}$ that only contain synchronised operations within S itself. The rationale behind these valid inequalities is the following. If tasks i and j of a given mandatory synchronised operation (i,j) are contained in set S, then $|\mathcal{U}_i| \leq |S| - 2$ and $|\mathcal{U}_j| \leq |S| - 2$ because i cannot be directly accessed from j and vice-versa, since synchronisation between i and j implies that these tasks must be performed in different routes. In these cases, and assuming that this logic applies to all other tasks in S, then the alternative to not being able to access a task from within S is from outside S, therefore increasing the number of arcs entering (and exiting) the set.

3.4.2 Strengthened capacity constraints

These valid inequalities cut infeasible solutions taking into account task capacity and synchronisation constraints. Considering sets $\mathcal{S}$ and S that were defined for previous inequalities, let us also consider an auxiliary set $S' = \{\{a,b\} : (a,b) \in \mathcal{R} \wedge a \in S \wedge b \in S\}$, which contains all the sets of synchronised tasks that are present in S. In parallel with previously defined $q(S) = \sum_{i \in S} e_i q_i$, let $q'(\mathcal{S}') = \sum_{\{a,b\} \in S'} (e_{ab} \min\{q_a, q_b\})$ designate the sum of the lightest demand in each mandatory synchronised operation. Under these assumptions, constraints (42) are valid:

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \max \left(2 \left\lceil \frac{q'(S')}{Q} \right\rceil, \left\lceil \frac{q(S)}{Q} \right\rceil \right) \tag{42}$$

3.4.3 Strengthened infeasible path constraints

Let path $P = (p_1, ..., p_{|P|})$ be an element of set $\mathcal{P}$, which represents the set of all infeasible paths. Furthermore, denote by $R(P)$ the set of all permutations of path P with length equal to $|P|$. Under these conditions, valid inequalities (43) can be introduced:

$$\sum_{i,j \in P} x_{ij} \leq |P| - 2 \qquad \forall P \subseteq \mathcal{P} : R(P) \subseteq \mathcal{P} \tag{43}$$

Constraints (43) are valid for paths that are infeasible regardless of the order that their constituting tasks are visited. In the context of the Robust VRPSync, they can be applied for paths that contain at least two tasks a and b that constitute an operation (a, b) that requires synchronisation. This is due to the fact that the synchronisation of an operation necessarily implies that a and b must be performed by different routes, meaning that there is no possible path that can be established between a and b in a feasible solution.

4 Solution approach

We develop a branch-and-cut algorithm for the Robust VRPSync. Current modern commercial solvers employ a branch-and-cut algorithm at their core. They also incorporate several advanced techniques and optimisations beyond the basic branch-and-cut framework, such as heuristics and presolve procedures. The user can complement these features with problem-tailored components. This is what we do in our approach.

The developed components are described in the following.

4.1 Pre-processing procedures

A set of pre-processing procedures was developed to eliminate redundancies, exploit the problem structure and reduce problem size.

Time window tightening In this procedure, parameters a_i and b_i, which are the lower and upper bounds of the time window of task i, respectively, are used to calculate tighter time window bounds dependent on the arcs that are traversed, denoted as a_{ij} and b_{ij}, i.e., a_{ij} is the earliest arrival time at task j when $(i, j) \in \mathcal{A}$ is traversed and b_{ij} represents the latest arrival time at task j when $(j, i) \in \mathcal{A}$ is traversed.

They are calculated as follows:

$$a_{ij} = \max\{a_j, a_i + s_i + d_{ij}\} \qquad \forall (i, j) \in \mathcal{A}$$
$$b_{ij} = \min\{b_j, b_i - d_{ji} - s_j\} \qquad \forall (j, i) \in \mathcal{A}$$

Considering the approach from Bertsimas and Sim (2004) and assuming a budget of uncertainty Γ for a given route, it is established that a robust-feasible route must be feasible for all combinations of Γ worst-case travel time deviations, regardless of the arcs on which these deviations will occur. This leads to the conclusion that, although a deviation may not necessarily materialise in a specific arc, there must be enough time slack to accommodate that deviation, otherwise, the route will not be robust-feasible. Considering this rationale, the time window tightening procedure was further expanded to consider the robust optimisation aspect of our problem.

Parameter $\hat{a}_{ij}$ gives the earliest arrival time at task j when $(i,j) \in \mathcal{A}$ is traversed and $\tilde{d}_{ij}$ takes its worst-case value. Analogously, parameter $\hat{b}_{ij}$ represents the latest arrival time at task j when $(j,i) \in \mathcal{A}$ is traversed and $\tilde{d}_{ji}$ takes its worst-case value.

$$\hat{a}_{ij} = \max\left\{a_j, a_i + s_i + d_{ij} + \hat{d}_{ij}\right\} \qquad \forall (i,j) \in \mathcal{A}$$
$$\hat{b}_{ij} = \min\left\{b_j, b_i - \hat{d}_{ji} - d_{ji} - s_j\right\} \qquad \forall (j,i) \in \mathcal{A}$$

These newly calculated parameters are then used in constraints (16) and (85) of models [RVRP-Sync1] and [RVRPSync3], respectively, to account for tighter time window intervals.

Arc elimination The previously computed tighter window bounds are used in an arc elimination step, applying the following rules for each arc $(i,j) \in \mathcal{A}$:

$$\begin{aligned} a_{ij} > b_j &\implies \text{arc } (i,j) \text{ is infeasible} \\ \hat{a}_{ij} > b_j \wedge \Gamma > 0 &\implies \text{arc } (i,j) \text{ is infeasible} \\ b_{ji} < a_i &\implies \text{arc } (i,j) \text{ is infeasible} \\ \hat{b}_{ji} > a_j \wedge \Gamma > 0 &\implies \text{arc } (i,j) \text{ is infeasible} \end{aligned}$$

After evaluating all of the arcs of the transportation network, the ones that were deemed infeasible are removed from set $\mathcal{A}$, consequently reducing the number of generated decision variables.

4.2 Implementation of valid inequalities

After performing the arc elimination procedure and instantiating the mathematical model, several valid inequalities are appended to the model directly.

For model [RVRPSync1], the entire routing network is combed through an enumeration procedure of consecutive arcs of set $\mathcal{A}$ to find small-size infeasible paths. This infeasible path identification procedure takes into account the robust optimisation aspects of the problem concerning the travel times between tasks. This is to be able to introduce valid inequalities based on infeasible paths. Iterating through each path $P = \left(p_1, ..., p_{|P|}\right)$ of length $|P|$, the arrival times are computed for each task in this path and checked for their feasibility within the established time window bounds. If at least one infeasibility is found, the tested path is infeasible and constraints (34) can be applied for that path, acting as valid inequalities. If the path is infeasible for all its possible path permutations, stronger valid inequalities (43) are applied instead.

In our solution approach, we only check for paths of length $|P| = 3$, since this procedure can be simplified by iterating through each path (i,j,l), with $(i,j),(j,l) \in \mathcal{A}$, and applying the following rules instead of having to compute the arrival times:

$$\begin{aligned} \Gamma \geq 2 \wedge \hat{a}_{ij} > \hat{b}_{lj} &\implies \text{path } (i,j,l) \text{ is infeasible} \\ \Gamma = 1 \wedge (\hat{a}_{ij} > b_{lj} \vee a_{ij} > \hat{b}_{lj}) &\implies \text{path } (i,j,l) \text{ is infeasible} \\ \Gamma = 0 \wedge a_{ij} > b_{lj} &\implies \text{path } (i,j,l) \text{ is infeasible} \end{aligned}$$

In model [RVRPSync2], infeasible path constraints do not act as valid inequalities because they are not redundant in this model and are essential to ensure model consistency. In this model, the constraints with $|P| = 3$ are already added at the root node and longer paths are separated and added to the model in a cutting plane fashion.

4.3 Separation routines and cutting planes

We develop the following cutting-plane separation routines, invoked during the optimisation process of the solver.

4.3.1 Subtour elimination

This separation routine is invoked whenever the solver finds a new candidate integer solution. For each identified subtour, and considering set S as containing all tasks involved in it, constraints (44) will be added to reject the candidate solution.

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \tag{44}$$

Since this separation routine is invoked for an already integer solution, subtour identification is trivial. Starting from arcs leaving the depot, the algorithm obtains the route sequences by sequentially combing through arcs whose solution value is equal to 1, until reaching the depot again. After obtaining all route sequences, the routine checks if any mandatory tasks ended up not being visited, in which case new subtours can be identified by sequentially combing through the arcs that start on those tasks.

4.3.2 Vehicle capacity separation

A separation routine is also implemented to generate cuts based on vehicle capacity constraints. To that effect, the CVRPSEP package is used. This separation routine can either be used to add cutting planes to increase the model's linear relaxation or to reject an infeasible candidate solution that violates capacity constraints.

For more information on the specific cutting planes of the CVRPSEP package, we refer the reader to Lysgaard, Letchford, and Eglese (2004).

4.3.3 Infeasible paths based on operation constraints

Given an integer candidate solution found by the solver, the solution is checked for operations (pairs of tasks) that should be performed by different routes but are instead being performed by the same route.

These infeasibilities are checked by iterating through the set of synchronised operations $\mathcal{R}$. For each operation (o, p) contained in the set, the routine checks if its tasks o and p are being performed in the same route, in which case the path that links these tasks, $(o, ..., p)$ or $(p, ..., o)$, is retrieved and established as infeasible.

For each identified path P, this routine will generate cutting planes (45), which will ultimately reject the newly-found solution.

$$\sum_{i,j \in P} x_{ij} \leq |P| - 2 \tag{45}$$

4.3.4 Infeasible paths based on time window bounds

If none of the above routines was able to identify any solution infeasibilities, the algorithm starts analysing task arrival times.

After retrieving a candidate solution, the infeasible path identification procedure described in Algorithm 1 is invoked for each of its routes. Table 3 visually exemplifies the procedure considering $\Gamma = 5$, which shows the calculation of all arrival times at tasks, i.e., the values of variables t_i^γ. This table also shows the calculation of two auxiliary vectors. The first vector takes value 1 if the arrival time of task i at Γ, t_i^Γ, is equal to the time window (TW) lower bound, a_i. This means that the arrival time of said task i did not depend on its preceding tasks; i.e., the preceding tasks had no impact on changing/increasing the arrival time of i. If $t_i^\Gamma > a_i$, the vector takes value 0. The second vector takes value 1 if the arrival time of task i at Γ, t_i^Γ, is lower or equal to the TW upper bound, b_i; 0, otherwise. In practice, this vector translates if the arrival times respect TW constraints and therefore represents whether the route is feasible or not.

Algorithm 1: Procedure for identifying infeasible paths based on time window bounds

input : route or path to be analysed, $(v_0, v_1, ..., v_r)$,
time window bounds, $[a_i, b_i]$,
task service times, s_i,
travel times and travel time deviations, d_{ij} and $\hat{d}_{ij}$,
budget of uncertainty, Γ,
arrival times, t_i^γ.

output: set of identified infeasible paths, S,
updated arrival times, t_i^γ.

1 $S := \varnothing; j := 0; f := -1; e := -1;$
2 $t_i^\gamma := 0, \forall i \in (v_0, v_1, ..., v_r), \gamma = 0, ..., \Gamma;$
3 **while** $j \leq r$ **do**
4 **if** $j = 0$ **then**
5 $t_{v_j}^\gamma := a_{v_j}, \forall \gamma = 0, ..., \Gamma;$
6 $f := 0;$
7 **else**
8 $t_{v_j}^0 := \max\left\{a_{v_j}, t_{v_{j-1}}^0 + s_{v_{j-1}} + d_{v_{j-1}v_j}\right\};$
9 $t_{v_j}^\gamma := \max\left\{a_{v_j}, t_{v_{j-1}}^\gamma + s_{v_{j-1}} + d_{v_{j-1}v_j}, t_{v_{j-1}}^{\gamma-1} + s_{v_{j-1}} + d_{v_{j-1}v_j} + \hat{d}_{v_{j-1}v_j}\right\}, \forall \gamma = 1, ..., \Gamma;$
10 **if** $t_{v_j}^\Gamma = a_{v_j}$ **then**
11 $f := j;$
12 **else if** $f \geq 0$ **and** $t_{v_j}^\Gamma > b_{v_j}$ **then**
13 $e := j;$
14 **append** $(v_f, ..., v_e)$ **to** $S;$
15 $f := -1; e := -1;$
16 $j := j + 1;$
17 **return** $S, t_i^\gamma;$

In addition to the instance problem parameters, the algorithm takes as input data a route or path $(v_0, ..., v_r)$ for which we intend to identify infeasible paths. As output data, it returns a set of identified infeasible paths within that route. The algorithm starts by setting all data structures and auxiliary variables to their initial states (lines 1–2). Auxiliary variables f and e designate the start and end positions of a possible infeasible path, respectively. Initially, these positions are indeterminate, and, therefore, $f = e = -1$. The algorithm iterates through each position j of the path and starts calculating the values of $t_{v_j}^\gamma$ (lines 5 and 8–9), according to the recursive definition of Munari et al. (2019). As the algorithm progresses through the path, f is updated with the most recent path position j where $t_{v_j}^\Gamma = a_{v_j}$, since this position marks the start of a possible infeasible path (lines 6 and 11).

On each position j of the path, the algorithm also checks if time windows are violated, i.e., if

Table 3: Infeasible path separation routine based on time window bounds ($\Gamma = 5$)

Route	**0**	→	**2**	→	**8**	→	**10**	→	**11**	→	**9**	→	**6**	→	**4**	→	**1**	→	**18**	→	**0**
a_i	0.0		50.0		120.0		100.0		75.0		155.0		350.0		620.0		605.0		522.0		0.0
b_i	925.0		412.0		630.0		876.0		390.0		310.0		650.0		765.0		836.0		925.0		925.0
Arrival Times, t_i^γ																					
$\gamma = 0$	0.0		50.0		131.0		172.0		247.0		302.0		352.0		620.0		695.0		790.0		884.0
$\gamma = 1$	0.0		50.0		146.0		187.0		262.0		317.0		367.0		620.0		708.0		808.0		902.0
$\gamma = 2$	0.0		50.0		146.0		192.0		275.0		330.0		380.0		620.0		708.0		821.0		920.0
$\gamma = 3$	0.0		50.0		146.0		192.0		280.0		338.0		388.0		620.0		708.0		821.0		933.0
$\gamma = 4$	0.0		50.0		146.0		192.0		280.0		343.0		395.0		620.0		708.0		821.0		933.0
$\gamma = 5$	0.0		50.0		146.0		192.0		280.0		343.0		400.0		620.0		708.0		821.0		933.0
$\boldsymbol{t_i^\Gamma = a_i}$	1		**1**		0		0		0		0		0		**1**		0		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1		1		1		**0**		1		1		1		1		**0**
Infeasible paths																					
No. 1			**2**	→	**8**	→	**10**	→	**11**	→	**9**										
No. 2															**4**	→	**1**	→	**18**	→	**0**

To compute t_i^γ, the following data was considered. Service time $s_i = 20.0$. Nominal travel times: $d_{0,2} = 24.0$; $d_{2,8} = 61.0$; $d_{8,10} = 21.0$; $d_{10,11} = 55.0$; $d_{11,9} = 35.0$; $d_{9,6} = 30.0$; $d_{6,4} = 62.0$; $d_{4,1} = 55.0$; $d_{1,18} = 75.0$; $d_{18,0} = 74.0$. Worst-case travel time deviations: $\hat{d}_{ij} = \lfloor 0.25 d_{ij} \rfloor$.

$t_{v_j}^\Gamma > b_{v_j}$. If that is the case and there is already a start position defined for a path (i.e., $f \geq 0$), an infeasible path has been identified, starting from position f up until position e. Therefore, path $(v_f, ..., v_e)$ will be added to the output of the algorithm and auxiliary variables f and e are re-set to their initial values (lines 13–15), to identify additional infeasible paths in the rest of the route.

Using the example provided in Table 3, the first task i where $t_i^\Gamma > b_i$ is task 9. Starting from task 9 backwards, the first task where $t_i^\Gamma = a_i$ is task 2. Therefore, the first infeasible path we are able to derive is 2–8–10–11–9. Starting from task 9, the next task where $t_i^\Gamma = a_i$ is task 4. Starting from this task, the next task where $t_i^\Gamma > b_i$ is task 0. Starting from this task backwards, the first task where $t_i^\Gamma = a_i$ is task 4. Therefore, the second infeasible path we are able to derive is 4–1–18–0.

With infeasible paths determined, we may be able to shorten them further by removing some of their initial tasks. This infeasible path minimisation procedure allows the algorithm to generate tighter infeasible path constraints with only the necessary arcs, thus avoiding the construction of redundant constraints throughout the algorithm. To that effect, we progressively shorten the path (i.e., progressively remove tasks from the beginning of the path) and apply the same algorithm described above. Table 4 exemplifies the shortening process of infeasible path #1. Considering sub-path 8–10–11–9, we set the arrival times of task 8 to the TW lower bound and calculate the subsequent arrival times.

If we verify that t_9^5 remains greater than b_9, then path 2–8–10–11–9 can be shortened to 8–10–11–9. By trying to shorten the path even further, we conclude that we are unable to obtain an infeasible path, because for path 10–11–9 the arrival time at task 9 already respects time window bounds. Therefore, 8–10–11–9 is a minimal infeasible path.

Having identified minimal infeasible paths, the separation routine adds cutting planes (46) for each infeasible path P.

$$\sum_{(i,j) \in A(P)} x_{ij} \leq |A(P)| - 1 \tag{46}$$

4.3.5 Infeasible paths based on a single synchronisation constraint

Separation routines were also developed to find infeasibilities in the arrival time matrix when a single synchronisation constraint is introduced. These routines are only invoked if, after invoking

Table 4: Minimisation of previously identified infeasible paths

Path	**8**	→	**10**	→	**11**	→	**9**
a_i	120.0		100.0		75.0		155.0
b_i	630.0		876.0		390.0		310.0
Arrival Times, t_i^γ							
$\gamma = 0$	120.0		161.0		236.0		291.0
$\gamma = 1$	120.0		166.0		249.0		304.0
$\gamma = 2$	120.0		166.0		254.0		312.0
$\gamma = 3$	120.0		166.0		254.0		317.0
$\gamma = 4$	120.0		166.0		254.0		317.0
$\gamma = 5$	120.0		166.0		254.0		317.0
$\boldsymbol{t_i^\Gamma = a_i}$	1		0		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1		**0**
Infeasible path							
	8	→	**10**	→	**11**	→	**9**

Path	**10**	→	**11**	→	**9**
a_i	100.0		75.0		155.0
b_i	876.0		390.0		310.0
Arrival Times, t_i^γ					
$\gamma = 0$	100.0		175.0		230.0
$\gamma = 1$	100.0		188.0		243.0
$\gamma = 2$	100.0		188.0		251.0
$\gamma = 3$	100.0		188.0		251.0
$\gamma = 4$	100.0		188.0		251.0
$\gamma = 5$	100.0		188.0		251.0
$\boldsymbol{t_i^\Gamma = a_i}$	1		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1
Feasible path					
	10	→	**11**	→	**9**

Algorithm 1 for all routes, it has not identified any infeasible paths. Using one synchronisation constraint at a time, the values of t_i^γ obtained from Algorithm 1 are updated to account for that single synchronisation constraint, according to the following rules:

Lower-bounded synchronisation (LB) Considering an operation $(o, p) \in \mathcal{R}$ subject to lower-bounded synchronisation with a time offset λ_{op}:

$$\text{if } t_o^{\gamma_1} + \lambda_{op} > t_p^{\gamma_2} \quad \Longrightarrow \quad t_p^{\gamma_2} := t_o^{\gamma_1} + \lambda_{op} \qquad \forall \gamma_1, \gamma_2 \in \mathbb{N}_0 : \gamma_1 + \gamma_2 = \Gamma$$

Upper-bounded synchronisation (UB) Considering an operation $(o, p) \in \mathcal{R}$ subject to upper-bounded synchronisation with a time offset μ_{op}:

$$\text{if } t_o^{\gamma_1} + \mu_{op} < t_p^{\gamma_2} \quad \Longrightarrow \quad t_o^{\gamma_1} := t_p^{\gamma_2} - \mu_{op} \qquad \forall \gamma_1, \gamma_2 \in \mathbb{N}_0 : \gamma_1 + \gamma_2 = \Gamma$$

The separation routine that identifies an infeasible group of paths based on a synchronisation constraint is described in Algorithm 2. Taking as input data the information of a given synchronisation constraint and the routes subject to it, the procedure starts by identifying the start of the paths that constitute a possible infeasible group of paths (lines 2–9), similarly to what happens in Algorithm 1. Afterwards, the values of t_i^γ, which were previously determined in Algorithm 1, are updated by applying the rules described above, according to the type of synchronisation constraint (lines 11 and 20). The changes to the arrival times of the synchronised tasks can induce changes to the tasks arrival times that follow, thus raising the need to recalculate the values of t_i^γ that follow task o, for upper-bounding synchronisation, or p, for lower-bounding synchronisation (lines 12–13 and 21–22). The algorithm then checks if any time window bounds are violated with the updated arrival times (lines 15–17 and 24–26), in which case an infeasible group of paths is identified and returned (lines 18 and 27).

Table 5 visually exemplifies the separation routine by considering two routes, 1 and 2, whose arrival times are represented. In this particular instance, a lower-bounded (LB) synchronisation constraint must be applied between tasks 7 and 8, with $\lambda_{7,8} = 50.0$.

We conclude that the synchronisation constraint is not being respected with these arrival times. By applying the rules above, we recalculate the arrival times of route 2, starting from task 8. This update procedure resulted in the arrival times that are represented in Table 6.

Algorithm 2: Procedure for identifying an infeasible group of paths based on a synchronisation constraint

input : first route subject to synchronisation, $(v_0, ..., o, ..., v_r)$,
second route subject to synchronisation, $(w_0, ..., p, ..., w_l)$,
operation subject to synchronisation, (o, p),
type of synchronisation constraint, type,
synchronisation time offset, λ_{op} or μ_{op},
time window bounds, $[a_i, b_i]$,
task service times, s_i,
travel times and travel time deviations, d_{ij} and $\hat{d}_{ij}$,
budget of uncertainty, Γ,
computed arrival times, t_i^γ.

output: infeasible group of paths, S,
updated arrival times, t_i^γ.

$S := \varnothing; f_1 := -1; e_1 := -1; f_2 := -1; e_2 := -1;$
$j := 0;$
while $j \leq r$ **do**
  if $t_{v_j}^\Gamma = a_{v_j}$ **then** $f_1 := j; j := j + 1;$
  else if $v_j = o$ **then** $e_1 := j;$ **break**;
$j := 0;$
while $j \leq l$ **do**
  if $t_{w_j}^\Gamma = a_{w_j}$ **then** $f_2 := j; j := j + 1;$
  else if $w_j = p$ **then** $e_2 := j;$ **break** ;
if type = "LB" **then**
  $t_p^\gamma := \max\left\{t_p^\gamma, t_o^{\Gamma-\gamma} + \lambda_{op}\right\};$
  $t_{w_j}^0 := \max\left\{t_{w_j}^0, t_{w_{j-1}}^0 + s_{w_{j-1}} + d_{w_{j-1}w_j}\right\}, \forall j = e_2 + 1, ..., l;$
  $t_{w_j}^\gamma := \max\left\{t_{w_j}^\gamma, t_{w_{j-1}}^\gamma + s_{w_{j-1}} + d_{w_{j-1}w_j}, t_{w_{j-1}}^{\gamma-1} + s_{w_{j-1}} + d_{w_{j-1}w_j} + \hat{d}_{w_{j-1}w_j}\right\}, \forall j = e_2 + 1, ..., l, \forall \gamma = 1, ..., \Gamma;$
  $j := e_2; e_2 := -1;$
  while $j \leq l$ **do**
    if $t_{w_j}^\gamma > b_{w_j}$ **then** $e_2 := j;$ **break**;
    $j := j + 1;$
  if $e_2 \geq 0$ **then** $S := \{(v_{f_1}, ..., v_{e_1}), (w_{f_2}, ..., w_{e_2})\};$
else if type = "UB" **then**
  $t_o^\gamma := \max\left\{t_o^\gamma, t_p^{\Gamma-\gamma} - \mu_{op}\right\};$
  $t_{v_j}^0 := \max\left\{t_{v_j}^0, t_{v_{j-1}}^0 + s_{v_{j-1}} + d_{v_{j-1}v_j}\right\}, \forall j = e_1 + 1, ..., r;$
  $t_{v_j}^\gamma := \max\left\{t_{v_j}^\gamma, t_{v_{j-1}}^\gamma + s_{v_{j-1}} + d_{v_{j-1}v_j}, t_{v_{j-1}}^{\gamma-1} + s_{v_{j-1}} + d_{v_{j-1}v_j} + \hat{d}_{v_{j-1}v_j}\right\}, \forall j = e_1 + 1, ..., r, \forall \gamma = 1, ..., \Gamma;$
  $j := e_1; e_1 := -1;$
  while $j \leq r$ **do**
    if $t_{v_j}^\gamma > b_{v_j}$ **then** $e_1 := j;$ **break**;
    $j := j + 1;$
  if $e_1 \geq 0$ **then** $S := \{(v_{f_1}, ..., v_{e_1}), (w_{f_2}, ..., w_{e_2})\};$
return $S, t_i^\gamma;$

Table 5: Illustration of the infeasible path separation routine based on synchronisation constraints

Route 1	**0**	→	**2**	→	**4**	→	**10**	→	**11**	→	**7***	→	**15**	→	**20**	→	**21**	→	**3**	→	**0**
a_i	0.0		75.0		255.0		357.0		410.0		532.0		422.0		727.0		635.0		533.0		0.0
b_i	995.0		412.0		630.0		876.0		634.0		600.0		733.0		831.0		855.0		995.0		995.0
Arrival Times, t_i^γ																					
$\gamma = 0$	0.0		75.0		243.5		357.0		411.0		543.1		639.0		727.0		820.5		929.1		979.1
$\gamma = 1$	0.0		75.0		243.5		357.0		416.5		544.6		641.7		727.0		822.0		938.4		988.4
$\gamma = 2$	0.0		75.0		243.5		363.6		420.0		546.1		646.7		727.0		823.5		939.9		989.9
$\gamma = 3$	0.0		75.0		265.0		368.0		431.5		546.1		646.7		727.0		825.0		941.4		991.4
$\gamma = 4$	0.0		75.0		265.0		368.0		434.5		546.1		646.7		727.0		826.1		942.9		992.9
$\gamma = 5$	0.0		75.0		265.0		373.5		439.0		546.1		646.7		727.0		827.2		944.0		994.0
$\boldsymbol{t_i^\Gamma = a_i}$	1		1		0		0		0		0		0		1		0		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1		1		1		1		1		1		1		1		1

Route 2	**0**	→	**24**	→	**6**	→	**12**	→	**13**	→	**8***	→	**18**	→	**22**	→	**25**	→	**0**
a_i	0.0		15.0		255.0		357.0		422.0		404.0		510.0		627.0		655.0		522.0
b_i	995.0		557.0		388.0		498.0		466.0		635.0		770.0		788.0		770.0		995.0
Arrival Times, t_i^γ																			
$\gamma = 0$	0.0		20.6		255.0		357.0		450.0		413.0		539.0		627.0		730.5		829.0
$\gamma = 1$	0.0		30.9		255.0		357.0		451.5		424.0		541.7		627.0		742.0		838.0
$\gamma = 2$	0.0		30.9		255.0		357.0		451.5		424.0		546.7		627.0		743.5		839.0
$\gamma = 3$	0.0		30.9		255.0		368.0		451.5		427.5		546.7		627.0		755.0		841.0
$\gamma = 4$	0.0		30.9		255.0		368.0		451.5		430.0		546.7		627.0		766.1		842.0
$\gamma = 5$	0.0		30.9		255.0		368.0		451.5		441.5		546.7		627.0		767.2		844.0
$\boldsymbol{t_i^\Gamma = a_i}$	1		0		1		0		0		0		0		1		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1		1		1		1		1		1		1		1

Table 6: Illustration of the infeasible path separation routine based on synchronisation constraints after arrival times are updated

Route 1	**0**	→	**2**	→	**4**	→	**10**	→	**11**	→	**7***	→	**15**	→	**20**	→	**21**	→	**3**	→	**0**
a_i	0.0		75.0		243.5		357.0		410.0		532.0		422.0		727.0		635.0		533.0		0.0
b_i	995.0		412.0		703.0		876.0		634.0		600.0		733.0		831.0		855.0		995.0		995.0
Arrival Times, t_i^γ																					
$\gamma = 0$	0.0		75.0		243.5		357.0		411.0		543.1		639.0		727.0		820.5		929.1		979.1
$\gamma = 1$	0.0		75.0		243.5		357.0		416.5		544.6		641.7		727.0		822.0		938.4		988.4
$\gamma = 2$	0.0		75.0		243.5		363.6		420.0		546.1		646.7		727.0		823.5		939.9		989.9
$\gamma = 3$	0.0		75.0		265.0		368.0		431.5		546.1		646.7		727.0		825.0		941.4		991.4
$\gamma = 4$	0.0		75.0		265.0		368.0		434.5		546.1		646.7		727.0		826.1		942.9		992.9
$\gamma = 5$	0.0		75.0		265.0		373.5		439.0		546.1		646.7		727.0		827.2		944.0		994.0
$\boldsymbol{t_i^\Gamma = a_i}$	1		1		0		0		0		0		0		1		0		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1		1		1		1		1		1		1		1		1
Infeasible path																					
			2	→	**4**	→	**10**	→	**11**	→	**7**										

Route 2	**0**	→	**24**	→	**6**	→	**12**	→	**13**	→	**8***	→	**18**	→	**22**	→	**25**	→	**0**
a_i	0.0		15.0		255.0		357.0		422.0		404.0		510.0		627.0		655.0		522.0
b_i	995.0		557.0		388.0		498.0		466.0		635.0		770.0		788.0		770.0		995.0
Arrival Times, t_i^γ																			
$\gamma = 0$	0.0		20.6		255.0		357.0		450.0		**596.1**		**678.0**		**703.0**		**780.0**		**829.0**
$\gamma = 1$	0.0		30.9		255.0		357.0		451.5		**596.1**		**687.5**		**715.3**		**783.0**		**838.0**
$\gamma = 2$	0.0		30.9		255.0		357.0		451.5		**596.1**		**687.5**		**715.3**		**783.0**		**839.0**
$\gamma = 3$	0.0		30.9		255.0		368.0		451.5		**596.1**		**687.5**		**715.3**		**799.0**		**841.0**
$\gamma = 4$	0.0		30.9		255.0		368.0		451.5		**596.1**		**687.5**		**718.0**		**799.0**		**842.0**
$\gamma = 5$	0.0		30.9		255.0		368.0		451.5		**596.1**		**687.5**		**722.3**		**799.0**		**844.0**
$\boldsymbol{t_i^\Gamma = a_i}$	1		0		1		0		0		0		0		0		0		0
$\boldsymbol{t_i^\Gamma \leq b_i}$	1		1		1		1		1		1		1		1		**0**		1
Infeasible path																			
					6	→	**12**	→	**13**	→	**8**	→	**18**	→	**22**	→	**25**		

The update procedure rendered an infeasibility in route 2, meaning that we can conclude that this current solution cannot enforce this synchronisation constraint. Therefore, we are able to retrieve a pair of infeasible paths that cannot occur simultaneously in the same solution.

For route 1, the one that was not updated, the infeasible path will extend from task 2 (the last task before task 7 where $t_i^\Gamma = a_i$) to task 7 (the synchronised task itself). For route 2, the one whose arrival times were updated, the infeasible path will extend from task 6 (the last task before task 8 where $t_i^\Gamma = a_i$) to task 25 (the task whose arrival time was rendered infeasible by the synchronisation constraint).

The cutting planes that result from this separation routine are an extension of the underlying principle of infeasible path constraints (43).

We consider that, if a set of paths $T = \{P_1, ..., P_{|T|}\}$ is used in a routing solution and it is deemed infeasible, then inequality (47) holds and can be introduced as a cutting plane:

$$\sum_{P \in T} \sum_{(i,j) \in A(P)} x_{ij} \leq \sum_{P \in T} |A(P)| - 1 \tag{47}$$

with $A(P)$ designating the set of arcs of path P. In the specific case of this separation routine, $|T|$ is equal to 2.

4.3.6 Infeasible paths based on multiple synchronisation constraints

The previous path separation routine considers only one synchronisation constraint at a time. However, there are some solutions whose infeasibility can only be verified when multiple synchronisation constraints are at play.

This separation procedure builds on the routine from Algorithm 2 and iteratively attempts to update the route arrival times after enforcing multiple synchronisation constraints. Table 7 shows a list of synchronisation constraints to be respected for a given problem instance.

Table 7: Illustration of the infeasible path separation routine based on multiple synchronisation constraints

#	Type	o	p	λ_{op}	μ_{op}	Task to update arrival times			Reference Task			Up to Date
						Task	Position	Route	Task	Position	Route	
7	LB	35	11	0.0	–	11	1	1	35	1	2	0
4	LB	45	31	0.0	–	31	1	3	45	5	4	0
5	LB	33	22	0.0	–	22	2	2	33	3	3	0
1	LB	15	42	0.0	–	42	2	4	25	5	1	0
6	UB	43	52	–	0.0	43	3	4	52	2	1	0
8	UB	14	27	–	0.0	14	4	1	27	4	2	0
2	UB	15	42	–	0.0	25	5	1	42	2	4	0
3	UB	45	31	–	0.0	45	5	4	31	1	3	0

Given that the synchronisation constraints that end up being selected in this separation routine depend on the concrete order of this list, the list can be shuffled a fixed number of times, in order to maximise the number of different infeasible combinations of synchronisation constraints.

The separation routine is executed as follows. Following the order of the previous table, Algorithm 2 is invoked for each synchronisation constraint, thus updating the values of t_i^γ. After updating, column "Up to Date" is changed to 1 for the respective constraint that was tested. If an infeasibility is found, the separation routine stops, and a new group of infeasible paths is found. If not, the separation routine progresses to the next synchronisation constraint.

However, depending on the order of the synchronisation constraints being tested, the updated values of t_i^γ may have had an impact on routes that were already tested previously with other synchronisation constraints. This means that it may be necessary to iterate multiple times through the constraints list, since updating certain synchronisation constraints may change arrival times where other synchronisation constraints are involved.

In the example shown in Table 7, it is possible to verify that, when Algorithm 2 is invoked for synchronisation constraint #1 ($\lambda_{15,42} = 0.0$), the arrival times of route #4, starting in position #2, are recalculated, as well as all subsequent tasks in the route. This means that synchronisation constraint #4 ($\lambda_{45,31} = 0.0$), which was checked before, will need to be re-checked, because one of its tasks (task 45) is also allocated to route #4, whose arrival times were just changed by constraint #1. As a consequence, column "Up to Date" will be changed back to 0 for constraint #4 and Algorithm 2 will need to be re-run for this constraint before proceeding through the remainder of the list.

If Algorithm 2 did not identify any infeasibilities and the procedure is able to get all of the constraints with its "Up to Date" column equal to 1 simultaneously, the candidate solution is feasible. However, if an infeasibility is found, the infeasible group of paths to consider is analogous to the one presented in Algorithm 2: for the route where the infeasibility was verified, the infeasible path will extend from the task where $t_i^\Gamma = a_i$ up to the task where $t_i^\Gamma > b_i$; for the remaining routes, the paths will extend from the task where $t_i^\Gamma = a_i$ up to their corresponding tasks subject to synchronisation.

The cutting planes applicable to this separation routine will correspond to constraints (47), applicable to each infeasible group of paths that was identified.

5 Computational experiments

A set of computational experiments was devised with the purpose of evaluating various indicators of the models and the solution approach. The instances used in these experiments and their detailed results can be found at `https://go.rfsoares.net/robustvrpsync`. The general conditions and assumptions of the conducted experiments are described below.

5.1 Test instances

The instances used in the computational experiments of this paper are based on the well-known instances from Solomon (1987), originally designed for the VRPTW. The Solomon instances are divided into three groups: 1) a group of instances whose generated customers are geographically clustered (group C); 2) instances where the locations of customers are randomly generated (group R); 3) instances where the locations of customers are both geographically clustered and randomly generated (group RC). The instances contain 100 customers. However, it is common practice to only consider a portion of the first customers for less tractable problems. For the problem at hand, only the first 25 customers were considered.

The adaptation of these instances to the VRPSync was performed as follows. In the process of generating the problem instances, three instance versions were considered with regard to synchronisation, namely: (i) instances without synchronisation (i.e., a standard VRPTW); (ii) instances with a subset of tasks subject to synchronisation where the synchronised tasks must be performed simultaneously; and (iii) instances with a subset of tasks subject to synchronisation with their corresponding arrival times subject to a given minimum and maximum offsets. Considering the different types of operation synchronisation in the literature, the instances where tasks are syn-

chronised simultaneously correspond to exact synchronisation (i.e., $\lambda_{op} = \mu_{op} = 0, \forall (o,p) \in \mathcal{R}$) and the instances with a minimum and maximum offset between synchronised tasks correspond to synchronisation subject to a minimum and maximum difference between arrival times (Dohn et al., 2011). In this latter case, the minimum and maximum time offsets considered were $\lambda_{op} = 0$ and $\mu_{op} = \min\{0.75 s_o, b_o - a_o\}, \forall (o,p) \in \mathcal{R}$. These established time offsets allow the instances to be used in any of the developed models, including model [RVRPSync1].

In terms of the number of tasks subject to synchronisation, it was established that 25% of the customers from the original Solomon instances should be subject to synchronisation. Therefore, for each of the instances, six customers are selected randomly, which will require two vehicles to be present at their locations. For these selected customers, their task nodes are duplicated to allow these customers to be visited by two vehicles. Then, the pairs of original and duplicate nodes will constitute synchronised operations and are added to the sets of synchronised operations, associated with the time offsets λ_{op} and μ_{op} that correspond to that instance version. The customers that were not selected for operation synchronisation remain unchanged.

Different levels of robustness were also considered in these instances. To that effect, different combinations of budgets of uncertainty and normalised deviations were considered as in Munari et al. (2019). The Γ values that were considered were 0 (deterministic), 1, 5 and 10. The different combinations of deviations and budgets of uncertainty from Munari et al. (2019) were adopted, thus constituting a total of 10 combinations. The Δ values that were considered were 0.10, 0.25, 0.50, resulting in a total number of 560 combinations of instances, budgets and deviations (10 per problem instance).

5.2 Evaluation of the solution approach

For evaluating the developed solution approach, only the set of instances subject to exact synchronisation was considered, since these instances have the most difficult synchronisation type, thus corresponding to the most adverse and constraining scenario. The computational experiments were performed on a machine with the Microsoft Windows 10 operating system, 64 GB of RAM and an AMD Ryzen 9 5950X @ 3.40 GHz CPU, with 16 cores and 32 threads. The commercial solver used for the experiments was Gurobi Optimizer 10.0. The experiments were performed in parallel, with each experiment capped to use a single processing thread and a maximum time limit of 3,600s.

5.2.1 Parameter settings

With the purpose of assessing the performance of different combinations of model versions, constraints and cut generation, six distinct parameter settings were created (A to F). They are summarised in Table 8 and are henceforth explained.

Parameter settings A and B consist of standalone solver approaches where models [RVRPSync3] (3-index) and [RVRPSync1] (2-index) are used, respectively. Apart from the maximum time limit, the solver is set to its default parameters and all essential constraints are implemented in the model.

Parameter settings C to F progressively build from the standalone solver approach by altering its standard branch-and-cut behaviour. They rely on the 2-index models and implement the previously described pre-processing procedures to improve the scalability of the solution process. For the infeasible path identification procedure included in the pre-processing phase, only paths of length 3 (i.e., 2-arc paths) were tested. Valid inequalities are also introduced into the models. For valid inequalities based on synchronisation constraints, these constraints are generated for all of their possible combinations of set S whose cardinality is between 2 and 3.

Table 8: Main characteristics of the different experiment versions

Setting	Model	Pre-proc	Constraints in initial model												Separated constraints				
											Valid inequalities				Relax	When new incumbent found			
			Base	Cap	Time	Sync	SEC	RCI	IPC	ISP	SEC1	SEC2	RCI	IPC	CapSep	SEC	CapSep	IPC	ISP
A	[RVRPSync3]	–	●	●	●	●	–	–	–	–	–	–	–	–	–	–	–	–	–
B	[RVRPSync1]	–	●	●	●	●	–	–	–	–	–	–	–	–	–	–	–	–	–
C	[RVRPSync1]	●	●	○	●	●	⋆	⋆	⋆	–	⋆	⋆	⋆	⋆	–	–	–	–	–
D	[RVRPSync1]	●	●	●	○	○	⋆	⋆	⋆	–	⋆	⋆	⋆	⋆	–	–	–	–	–
E	[RVRPSync1]	●	●	–	○	○	⋆	⋆	⋆	–	⋆	⋆	⋆	⋆	●	–	–	–	–
F	[RVRPSync2]	●	●	–	–	–	◇	◇	◇	–	◇	◇	◇	◇	–	●	●	●	●

Legend: Pre-proc: Pre-processing procedures (arc elimination and time window tightening). **Constraints in initial model:** Base: Constraints (8)–(11) and (23)–(26) (Model [RVRPSync1]), constraints (28)–(31) and (36) (Model [RVRPSync2]), constraints (77)–(80) and (94)–(97) (Model [RVRPSync3]); Cap: Capacity constraints (12)–(13) (Model [RVRPSync1]), constraints (81)–(82) (Model [RVRPSync3]); Time: Arrival time and time window constraints (14)–(16) (Model [RVRPSync1]), constraints (83)–(85) (Model [RVRPSync3]); Sync: Synchronisation constraints (17)–(22) (Model [RVRPSync1]), constraints (86)–(93) (Model [RVRPSync3]); SEC: Strengthened subtour elimination constraints (32); RCI: Rounded capacity constraints (33); IPC: Infeasible path constraints (34); ISP: Infeasible sets of paths constraints (35); ***Valid inequalities:*** SEC1: Strengthened subtour elimination constraints (37)–(39); SEC2: Strengthened subtour elimination constraints (40)–(41); RCI: Strengthened capacity constraints (42); IPC: Strengthened infeasible path constraints (43). ●: Implemented; ○: Implemented as lazy constraints; ◇: Implemented for sets S with cardinality up to 3 (SEC, SEC1, SEC2, RCI) or paths with length 3 (IPC); ⋆: Implemented as lazy constraints for sets S with cardinality up to 3 (SEC, SEC1, SEC2, RCI) or paths with length 3 (IPC). **Separated constraints:** ***Relax:*** when new relaxed solution found; CapSep: Constraints generated by the capacity separation routine of the CVRPSEP library; SEC: Subtour elimination constraints (44); IPC: Infeasible path constraints (45)–(46); ISP: Infeasible sets of paths constraints (47).

Setting C implements capacity constraints and valid inequalities as lazy constraints, from which certain constraints from the [RVRPSync2] model are equally implemented with this purpose. This implementation alters the solver's behaviour in the branch-and-cut process since the lazy constraints will only be added if a candidate solution violates them. This approach is common for branch-and-cut approaches and is typically applied for sets of constraints whose size increases exponentially with larger instance sizes. The management of these lazy constraints is handled by the solver, requiring no additional implementation by the user.

Setting D differs from experiment version C by implementing all time-related constraints as lazy constraints. These time-related constraints encompass all constraints where decision variable t_i^γ is involved. Capacity constraints, on the other hand, are implemented as traditional constraints.

Settings E and F further customise the solver's optimisation process by removing certain constraints from the initial model and by generating them in a cutting plane fashion.

In setting E, capacity constraints are removed from the initial model. Whenever the solver obtains a new relaxed solution, the separation routine of the CVRPSEP package is invoked.

Finally, in setting F, the initial model is stripped of all capacity and time-related constraints. Valid inequalities are implemented as traditional constraints to account for part of the effect of the constraints that were removed. Whenever the solver is able to find a new candidate solution, the separation routines described in the previous section are run sequentially in the order they are presented. If a separation routine identifies an infeasibility, the corresponding cutting planes are added to the model, and the following routines are skipped for that candidate solution.

5.2.2 General performance evaluation

The obtained results for each experiment version and instance group are summarised in Table 9. This table presents four key performance indicators: the number of instances that were able to obtain a solution, the number of instances that were determined to be infeasible by the solver, the number of instances that were solved optimally, the average of the obtained optimality gaps and their standard deviation.

In a general analysis, it is possible to observe that the more easily solvable instance groups are the clustered ones (groups C1 and C2), followed by the randomly generated ones (groups R1 and R2), due to the high number of instances that were able to be solved optimally. This behaviour appears to remain constant regardless of the experiment versions considered.

With the 3-index model (parameter setting A), only 27 instance-parameter combinations could be solved (out of 560). Furthermore, the commercial solver is unable to find a feasible solution for 477 of these instances. When compared with the other experiment versions, the average gaps for this experiment version A are also significantly higher. They are more than double the ones observed in the remaining experiments. This last indicator, along with the high variability of the optimality gaps, clearly demonstrates and confirms the lack of tractability of the general 3-index model. This observation is consistent with other reports from the literature (e.g., Agra et al., 2012; Lee et al., 2012) when using robust optimisation models with 3-index routing formulations.

Experiment versions B to F, which use the 2-index models, behave more favourably. In these experiments, for all settings either a feasible solution could be found or model infeasibility was proven. Instance groups RC1 and RC2 continue to be the hardest to solve, given their high average gaps. Nevertheless, the models for these experiments appear to be more tractable, since the number of optimally solved instances ranges from 262 (for experiment version B) to 302 (for experiment version D). The variability of the optimality gaps does not vary significantly across these experiment versions since the standard deviation of the optimality gaps range from 10.6%

Table 9: Performance indicators for different instance groups and experiment versions

Group	Indicator	Version					
		A	B	C	D	E	F
C1	# sol	34	90	90	90	90	90
	# inf	0	0	0	0	0	0
	# opt	15	62	66	67	66	61
	gap, avg	13.7%	2.8%	3.0%	2.8%	2.7%	3.6%
	gap, std	25.7%	5.7%	6.2%	5.9%	5.7%	7.2%
C2	# sol	14	80	80	80	80	80
	# inf	0	0	0	0	0	0
	# opt	10	58	60	60	60	60
	gap, avg	19.5%	4.6%	5.2%	4.3%	5.0%	5.1%
	gap, std	32.7%	8.3%	9.5%	7.9%	8.9%	9.1%
R1	# sol	15	108	108	108	108	108
	# inf	9	12	12	12	12	12
	# opt	2	58	60	64	64	62
	gap, avg	5.9%	5.5%	5.1%	4.5%	4.5%	4.7%
	gap, std	7.5%	7.6%	7.4%	6.9%	6.8%	7.1%
R2	# sol	4	110	110	110	110	110
	# inf	0	0	0	0	0	0
	# opt	0	41	43	51	47	46
	gap, avg	46.5%	7.1%	6.9%	5.7%	6.2%	6.5%
	gap, std	21.7%	7.1%	7.0%	6.6%	6.8%	7.1%
RC1	# sol	3	77	77	77	77	77
	# inf	3	3	3	3	3	3
	# opt	0	23	30	33	34	29
	gap, avg	17.3%	8.2%	8.0%	7.1%	7.1%	7.7%
	gap, std	12.0%	7.9%	8.4%	7.7%	7.6%	7.7%
RC2	# sol	1	80	80	80	80	80
	# inf	0	0	0	0	0	0
	# opt	0	20	20	27	25	26
	gap, avg	68.5%	20.8%	20.7%	19.6%	20.7%	20.6%
	gap, std	–	16.3%	16.3%	17.2%	17.5%	17.5%
Total	# sol	71	545	545	545	545	545
	# inf	12	15	15	15	15	15
	# opt	27	262	279	302	296	284
	gap, avg	16.0%	7.9%	7.8%	7.0%	7.4%	7.7%
	gap, std	25.8%	10.8%	10.9%	10.6%	11.0%	11.1%

Legend: Group: Solomon instance group; # sol: number of tests that were able to obtain a solution; # inf: number of tests identified as infeasible; # opt: number of tests solved to optimality; gap, avg: average optimality gap; gap, std: standard deviation of the optimality gaps.

(for experiment version D) to 11.1% (for experiment version F).

It is interesting to see that higher degrees of customisation in the solution approach appear to be counterproductive. Specifically, setting F, which removes several problem constraints from the initial model and separates infeasible path constraints instead, appears not to yield better results than approaches simply resorting to lazy constraints, such as parameter setting D.

Overall, setting D appears to be the most suitable approach, since it exhibits the lowest average gap and standard deviation value. Furthermore, it is able to optimally solve the most instances, which corresponds to approximately 55% of all instance-parameter combinations.

In an attempt to determine the differences in solution strategies between distinct experiment versions, Figure 1 presents scatter plots that compare each model's final linear relaxation and solution improvements, using experiment version B as a basis for comparison. To increase the readability of these plots, instances that were solved optimally in both experiment versions were removed.

The plots show a great concentration of points around the y-axis, meaning that a significant number of instances observe a solution improvement equal (or approximately equal) to zero. This conclusion could be explained by either the difficulty of this approach in finding new incumbent solutions or by the fact that the solver is already able to find (near-)optimal solutions, hence the low or near-zero solution improvement. After additional analysis of these data points, it is verifiable that these data points correspond to instances with low optimality gaps.

When analysing the dispersion of the solution improvement indicator between instance groups, the percentual improvements are rather heterogeneous. However, it is clearly visible that instance group RC2 exhibits the most inconsistent improvement values. Outliers are typically attributed to instances with larger budgets of uncertainty. This observation is verifiable across all experiment versions, although this variability is more expressive in experiment versions D and F, ranging from approximately -12% to 2% and from -15% to 4%, respectively. On the other hand, instance group RC1 appears to be the most consistent, as the solution improvement indicator typically ranges from -3% to 4%.

Nevertheless, in a general analysis, all experiment versions provide a net benefit in terms of solution improvement. In fact, the number of instances where solution values improved or maintained the same amount to 60%, 62%, 52% and 49% for experiment versions C, D, E and F, respectively. Regarding the Best Bound Improvement, we also observe a general increase in this indicator. In fact, the models that improved their linear relaxation amount to 64%, 88%, 83% and 71% for experiment version C, D, E and F, respectively.

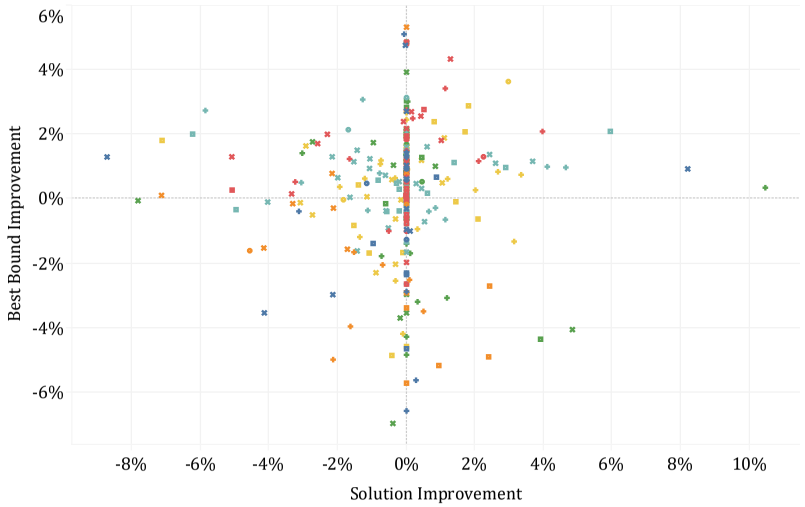
In sum, these results suggest that the developed solution approach, when compared to a standalone solver approach, has a positive effect on both the strengthening of the final linear relaxation of most models and do not generally appear to render worse solutions.

5.2.3 Performance comparison between different levels of robustness

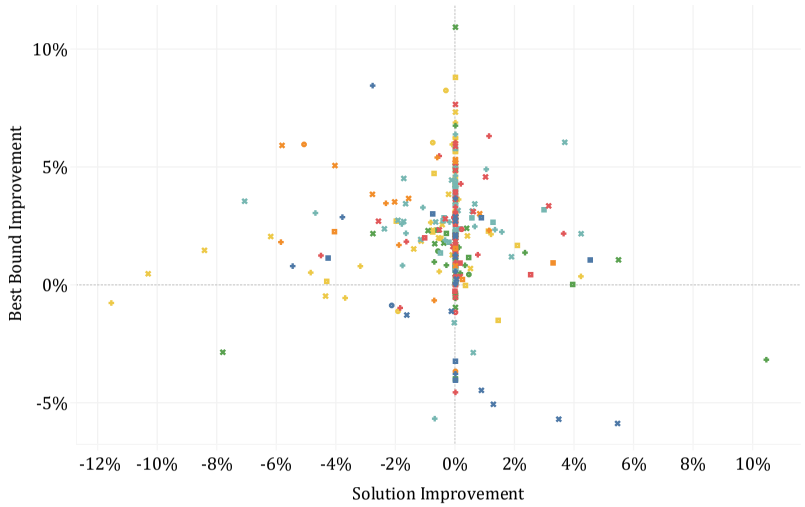
To detect potential differences in performance triggered by the degree of robustness of the instances, Table 10 presents the same results disaggregated by budget of uncertainty and deviation values.

The 2-index deterministic models (i.e., $\Gamma = 0$) appear to have the best performance, since the variability of the optimality gaps and their average tend to be lower than in robust instances, where the average gaps and standard deviations tend to be up to three and two percentage points higher, respectively. This is not the case for the solution approaches relying on the 3-index model (parameter setting A).

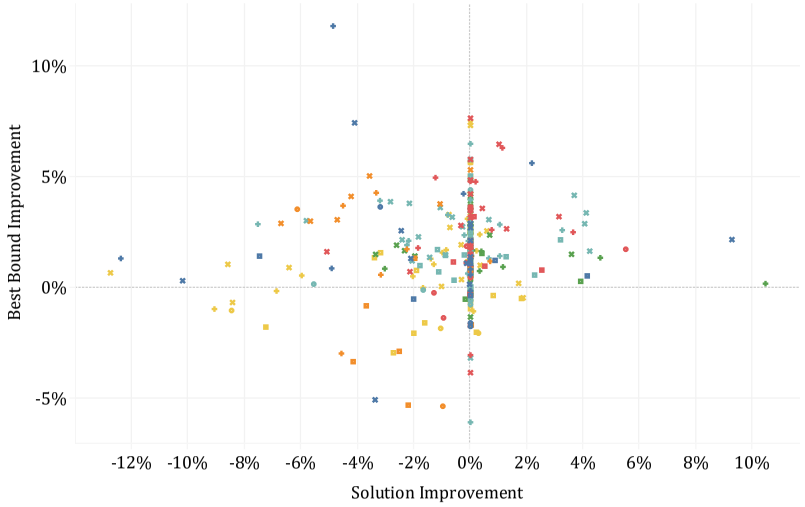
When comparing different deviation values within the same budget of uncertainty, no significant



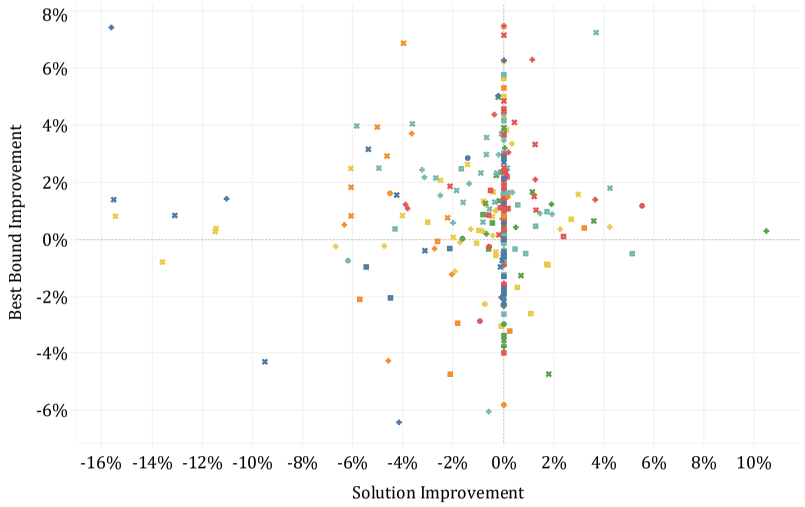
(a) Experiment version C



(b) Experiment version D



(c) Experiment version E



(d) Experiment version F

Figure 1: Best Bound and Solution Improvement indicators (comparison basis: experiment version B)

Table 10: Performance indicators for different experiment versions, budgets of uncertainty and deviations

Version	Indicator	$\Gamma = 0$	$\Gamma = 1$			$\Gamma = 5$			$\Gamma = 10$		
		$\Delta = 0.00$	$\Delta = 0.10$	$\Delta = 0.25$	$\Delta = 0.50$	$\Delta = 0.10$	$\Delta = 0.25$	$\Delta = 0.50$	$\Delta = 0.10$	$\Delta = 0.25$	$\Delta = 0.50$
A	# sol	9	10	8	8	5	6	6	7	6	6
	# inf	0	0	0	4	0	0	4	0	0	4
	# opt	4	4	2	3	3	2	2	3	2	2
	gap, avg	15.1%	13.1%	9.1%	10.0%	14.5%	14.4%	15.7%	24.8%	24.1%	23.9%
	gap, std	25.1%	18.6%	18.2%	20.1%	31.1%	27.8%	27.6%	33.4%	36.4%	35.7%
B	# sol	56	56	56	51	56	56	51	56	56	51
	# inf	0	0	0	5	0	0	5	0	0	5
	# opt	30	29	29	28	25	26	28	22	23	22
	gap, avg	6.4%	6.6%	7.0%	7.1%	8.0%	8.0%	8.5%	9.1%	8.8%	9.3%
	gap, std	9.9%	10.1%	10.4%	10.6%	10.6%	11.1%	11.7%	10.9%	11.0%	11.9%
C	# sol	56	56	56	51	56	56	51	56	56	51
	# inf	0	0	0	5	0	0	5	0	0	5
	# opt	31	31	30	28	27	29	29	22	27	25
	gap, avg	6.1%	6.7%	7.0%	7.3%	8.0%	8.0%	8.3%	9.0%	8.9%	9.3%
	gap, std	9.7%	10.2%	10.4%	10.8%	10.8%	11.2%	11.9%	11.1%	11.6%	12.0%
D	# sol	56	56	56	51	56	56	51	56	56	51
	# inf	0	0	0	5	0	0	5	0	0	5
	# opt	34	33	33	30	30	31	29	26	29	27
	gap, avg	5.7%	5.8%	5.9%	6.6%	7.2%	7.2%	7.6%	8.1%	7.7%	8.5%
	gap, std	9.5%	9.7%	9.8%	10.3%	10.6%	11.2%	11.3%	11.3%	11.2%	11.8%
E	# sol	56	56	56	51	56	56	51	56	56	51
	# inf	0	0	0	5	0	0	5	0	0	5
	# opt	32	32	33	30	29	29	28	25	31	27
	gap, avg	6.5%	6.4%	6.5%	7.2%	7.5%	7.2%	7.9%	8.0%	7.9%	8.9%
	gap, std	10.4%	10.5%	10.5%	11.0%	11.2%	10.9%	11.6%	10.9%	11.2%	12.2%
F	# sol	56	56	56	51	56	56	51	56	56	51
	# inf	0	0	0	5	0	0	5	0	0	5
	# opt	30	30	33	28	27	31	27	24	28	26
	gap, avg	6.6%	6.5%	6.8%	7.4%	7.5%	7.7%	8.4%	8.9%	8.3%	9.1%
	gap, std	10.1%	10.1%	10.5%	11.0%	10.7%	11.4%	11.8%	12.1%	11.8%	12.2%

Legend: Version: Experiment version; # sol: number of tests that were able to obtain a solution; # inf: number of tests identified as infeasible; # opt: number of tests solved to optimality; gap, avg: average optimality gap; gap, std: standard deviation of the optimality gaps.

differences in performance can be observed. However, the number of models solved to optimality appears to increase as deviation values also increase. This is indicative that the solution approach appears to have better performance with tighter constraints.

When performing comparisons between different budget values with the same deviation value, we observe that the average gaps increase by an order of one percentage point between $\Gamma = 1$ and $\Gamma = 5$ and another percentage point between $\Gamma = 5$ and $\Gamma = 10$. This general gap increase, however, is not verified in terms of variability, since the standard deviation remains stable. However, the number of optimally solved instances appears to decrease with larger budgets of uncertainty, which is very likely a consequence of the increased model sizes as Γ values increase.

In conclusion, it is possible to state that robustness appears to have a non-negligible, albeit limited impact on the performance of the adopted solution approaches. A higher budget of uncertainty has a slight, although non-negligible impact on the performance of the solution approach, while it is verifiable that higher deviation values allow the solution approach to converge more easily, since it appears to further limit the solution space, where the solver tends to behave more favourably.

5.3 Comparison of routing solutions

To evaluate the impact of synchronisation and robust optimisation on the obtained routing solutions, the solution structures were analysed and two indicators were calculated to enable comparisons between model runs in an aggregate manner. The first indicator consists of the percentual variation ("% diff.") of the value of the objective function between two pairs of models. The second indicator, for which the name "route similarity ratio" was adopted, consists of the ratio between the number of routing arcs that remained unchanged from one solution to another and the total

number of distinct arcs that are traversed in both solutions.

These two ratios were calculated between all combinations of budgets of uncertainty, as well as deviation values and instance versions (i.e., between instances with/without synchronisation).

For performing these comparisons in the most reliable way possible, only nearly-optimal solutions were considered, and only comparisons for model [RVRPSync1] were calculated due to it being the most tractable model. For the purpose of this paper, a solution is considered to be nearly-optimal if its corresponding model run obtained an optimality gap that is not higher than 5.0%, which corresponds to 310 instances out of the 560 instance-parameter combinations.

To expand the scope of these comparisons and adequately perform these analyses, the 2-index models without synchronisation were run with the commercial solver for a period of 3,600s. Furthermore, all problem instances were re-run adopting a robust optimisation approach similar to the one proposed by Soyster (1973). In this alternative approach, which can be deemed "ultra-conservative", it is assumed that an uncertain parameter's worst-case scenario always occurs. In the problem at hand, this adaptation is achieved by re-generating all deterministic models such that parameter $d_{ij} := d_{ij} + \hat{d}_{ij}$.

The results of both the deterministic version and the robust approach by Soyster (1973) ultimately provide us with lower and upper-bounds on the impacts of travel time uncertainty in a robust optimisation setting.

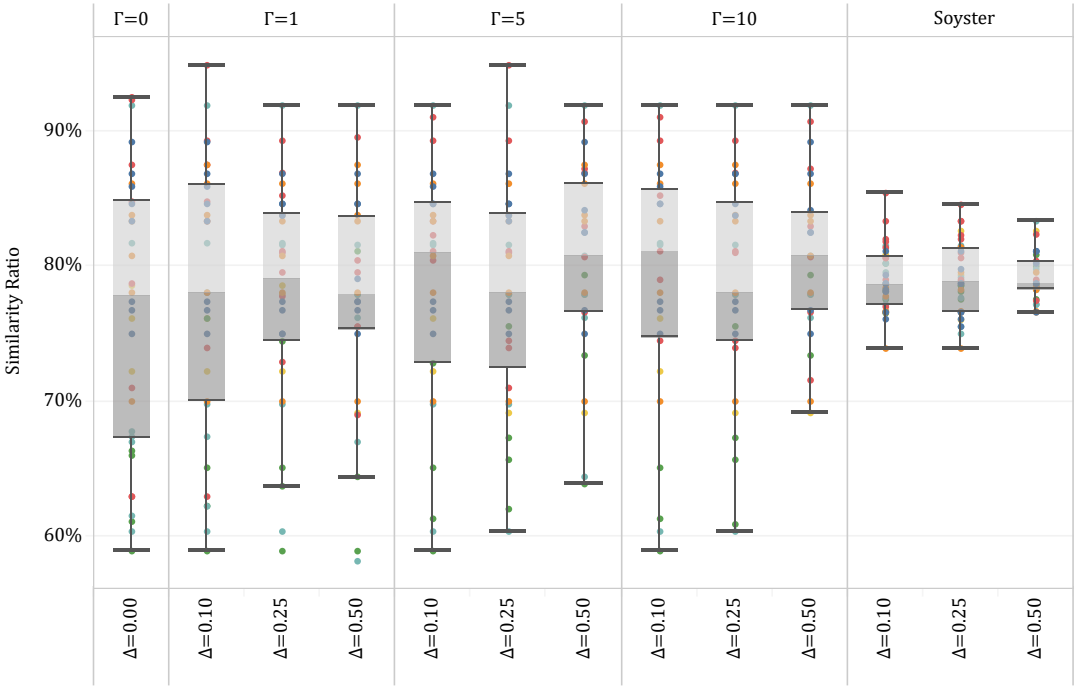
5.3.1 Effects of synchronisation

To allow for a more direct comparison of the impacts of synchronisation in the routing plans, Figure 2 represents the dispersion of the route similarity ratio and the objective function percentual difference for the model versions with exact synchronisation, using as a basis for comparison the model versions without synchronisation.

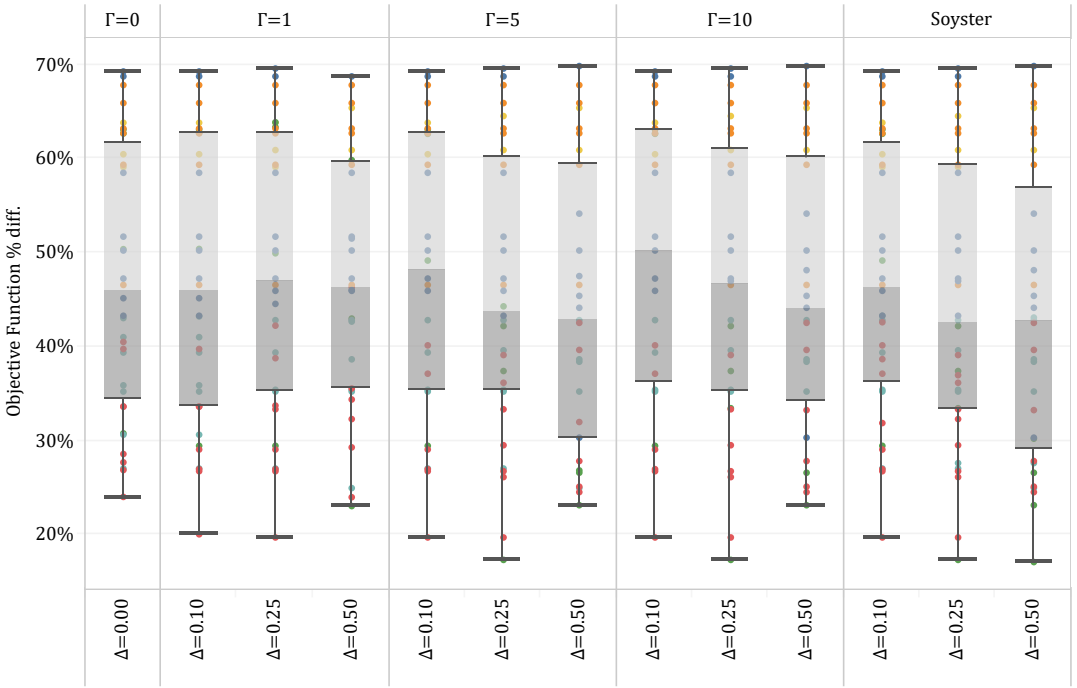
The figure shows that the solutions of the models with exact synchronisation resort to at least 55% of the same arcs present in the solution from the models without synchronisation. The similarity ratio seems to decrease its variability as budgets of uncertainty and normalised deviations also increase, thus creating solutions that are apparently more similar to the ones found in the model versions without synchronisation. This is especially visible in the Soyster models, where the variability decreases significantly. This phenomenon may occur because synchronisation constraints may induce similar effects to the ones induced by an increased level of robustness of the solution. This may be a possible explanation since the introduction of synchronisation constraints usually induces an increase in vehicle idle times, which is also an observed effect when considering worst-case travel times in the context of robust optimisation.

With regard to the percentual difference of the objective function values, the box plots exhibit similar dispersion of the indicator between different budgets of uncertainty and deviation values. This indicator, however, can vary significantly from 17% to 70%. It is noteworthy that group R2 appears to be the instance group with the least impact on the objective function, and group C2 appears to have its instances impacted the most. This observation may be justified by the instance groups underlying design and their customers' geographical distribution. In fact, when acknowledging synchronisation constraints, instances that contain clusters of customers may require routes to travel further distances to ensure these constraints are met. Although this is also valid for instances with randomly distributed customers, this effect is less expressive since the opportunity cost of visiting another customer instead tends to be lower.

Complementing these analyses, the instances subject to synchronisation with minimum and maximum difference were also tested and their results were compared with the ones obtained



(a) Route Similarity Ratio, exact synchronisation



(b) Objective Function % diff., exact synchronisation

Figure 2: Dispersion of the comparison indicators for exact synchronisation (comparison basis: instances without synchronisation)

with exact synchronisation. The purpose of this comparison was to assess the existence of any significant difference between the solution structures of these two types of synchronisation. Figure 3 represents the dispersion of the comparison indicators for these instances, using as a basis the instances subject to exact synchronisation.

The results show that the instance group with the most propensity to adopt different solution structures is group C1, since this group corresponds to the one that exhibits the most similarity ratios lower than 100%. This can probably be justified by the fact that this particular instance group has the highest service time for each customer, making parameters μ_{op} take the highest values among all groups. This allows the problem to accommodate more distinct solutions since the synchronisation constraints are less restrictive and, therefore, the solution space is larger.

Nevertheless, the results also suggest that the impacts of a less strict type of synchronisation are limited, since the route similarity ratio is equal to 100% for at least 50% of the instances. This means that, for more than half of the instances, there are no changes to solution structure between the adoption of exact synchronisation constraints and the adoption of minimum and maximum difference between the arrival times of the synchronised tasks. This is also confirmed by the percentual differences of the objective function, since we observe very low dispersion in this indicator, with most of its values around 0%.

In sum, the comparisons between exact synchronisation and synchronisation with a minimum and maximum difference between synchronised tasks suggest that there are no significant changes in solution behaviour. However, it should be emphasised that the underlying limitations of the generated instances, namely the adopted time offset values between the synchronised tasks, may be limiting these conclusions.

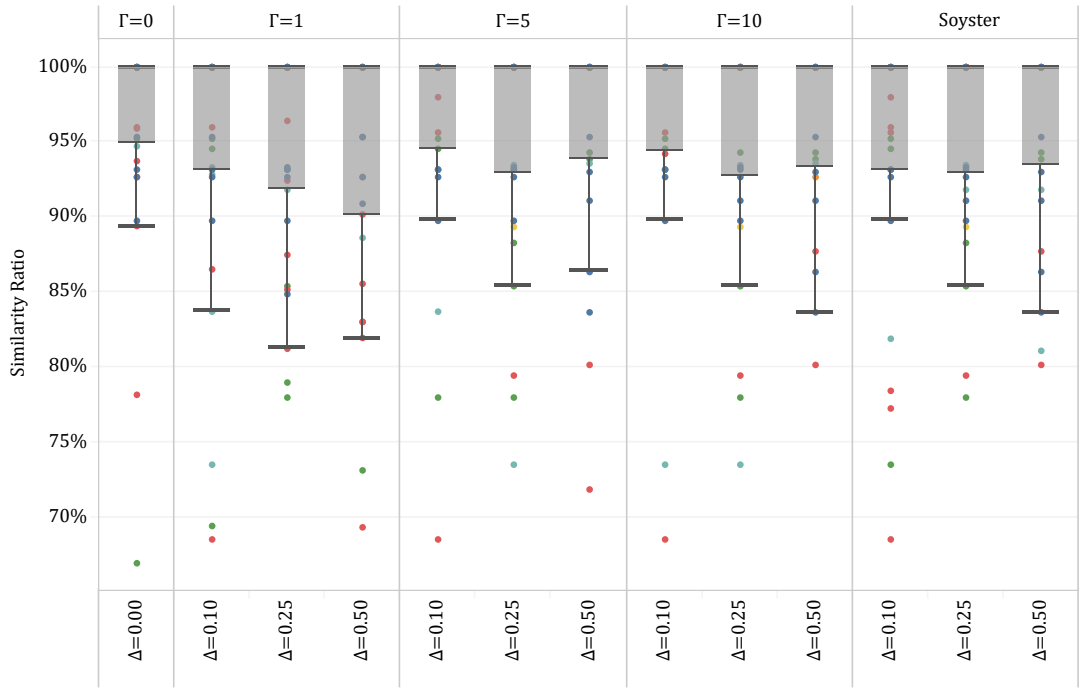
5.3.2 Effects of robustness

Figure 4 and its sub-figures present box and whiskers plots representing the dispersion of the route similarity ratio and the objective function percentual difference, using the deterministic models ($\Gamma = 0, \Delta = 0$) as a basis for the calculation of the indicators. Specifically, Figures 4a and 4b present the dispersion of the percentual variation of the objective function for different deviation values, while Figures 4c and 4d present the dispersion of the route similarity ratio.

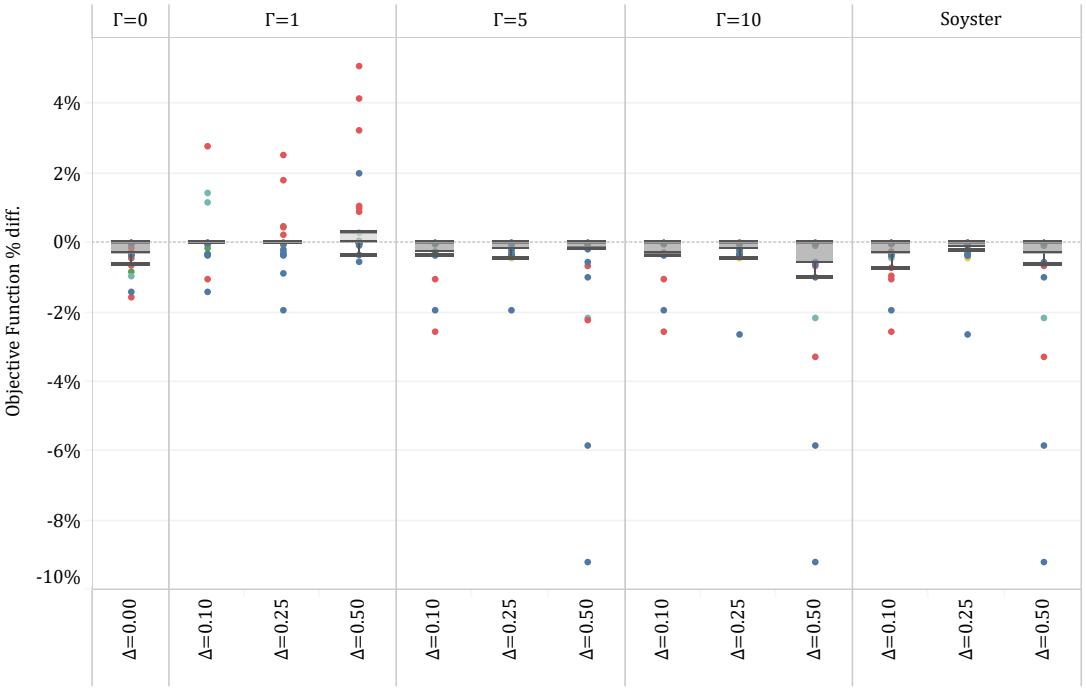
In these comparisons, the objective function percentual difference indicator quantifies the difference between robust scenarios and their deterministic counterparts and is therefore equivalent to the concept of Price of Robustness (PoR), as defined by Gorissen, Yanıkoğlu, and den Hertog (2015). The purpose of the PoR is to quantify how much robustness leaves on the table in terms of the value of the objective function.

In both indicators, it is possible to observe the increase in the dispersion of this ratio as the Γ and Δ values increase, as would be expected. For the % diff. indicators, most occurrences take values close to 0%. For the route similarity ratio, similar conclusions can be taken, as more than 50% of the occurrences of this ratio take values greater than 90%. This suggests that the impact of robust optimisation on the routing plans is strongly dependent on the specific instance.

Nevertheless, the figures show that there is a clear increase in the dispersion of both indicators when transitioning from $\Gamma = 1$ to $\Gamma = 5$. However, there is hardly any difference between the dispersion values of $\Gamma = 5$ and $\Gamma = 10$, as well as with the Soyster models. This suggests that the marginal price of considering a more robust setting beyond $\Gamma = 1$ is negligible, with limited impact on the solution structure and objective function values. After further analysis, it was verified that, for instances of this size, there are hardly any routes performing more than ten tasks, meaning that, in these solutions, the routes will consider the travel time deviation in all traversed arcs, thus

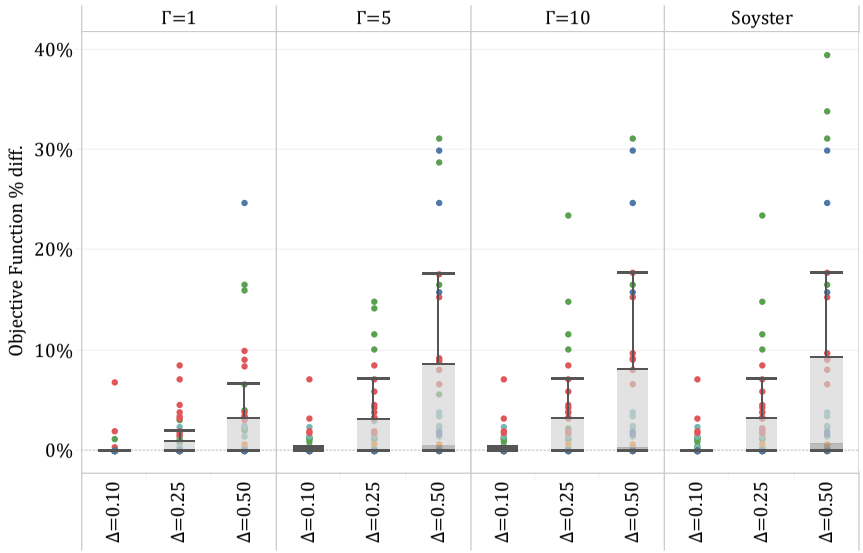


(a) Route Similarity Ratio, minimum + maximum difference

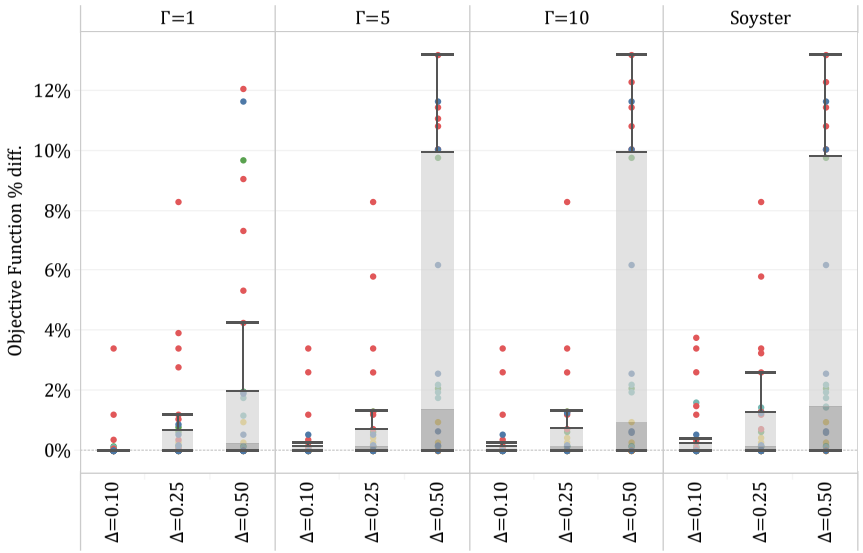


(b) Objective Function % diff., minimum + maximum difference

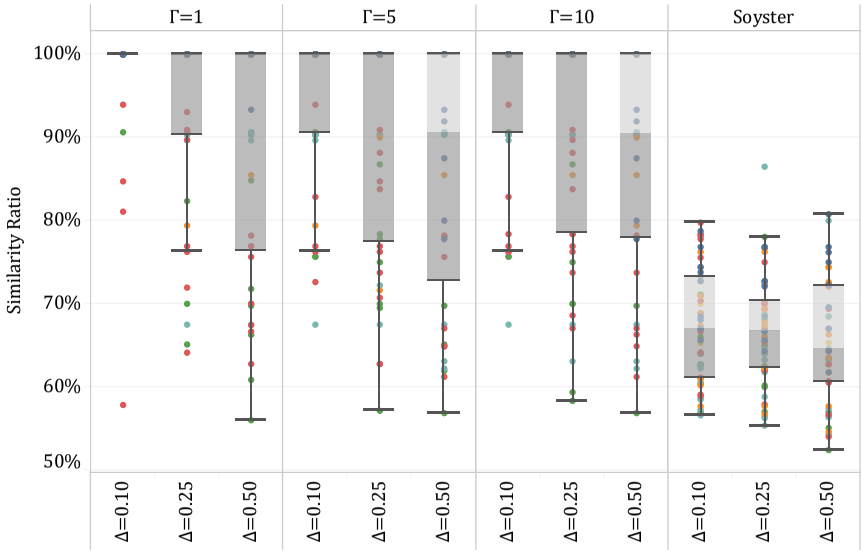
Figure 3: Dispersion of the comparison indicators for synchronisation with minimum + maximum difference (comparison basis: instances with exact synchronisation)



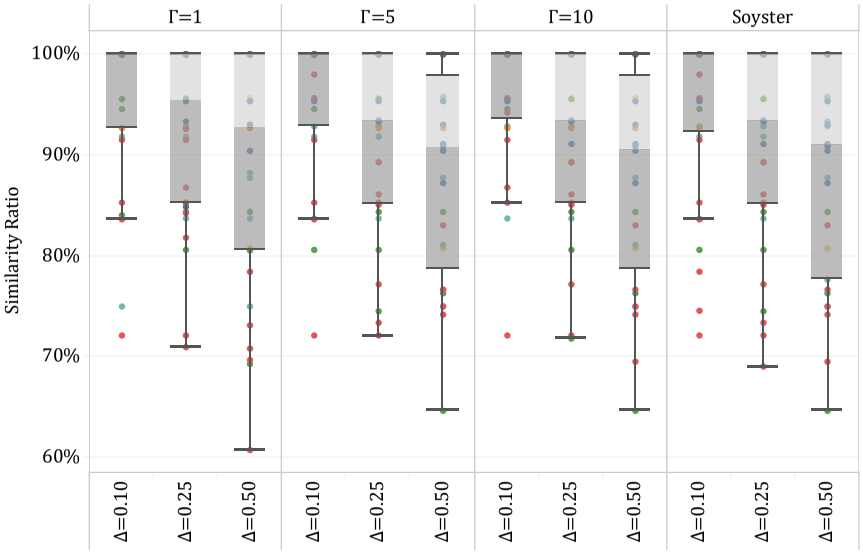
(a) Objective Function % diff., no synchronisation



(b) Objective Function % diff., exact synchronisation



(c) Route Similarity Ratio, no synchronisation



(d) Route Similarity Ratio, exact synchronisation

Legend: Instance Groups C1 C2 R1 R2 RC1 RC2

Figure 4: Visual representation of the dispersion of comparison indicators (comparison basis: $\Gamma = 0, \Delta = 0$)

equating these solutions to the ones obtained in the Soyster models.

Another observation that can be taken from the analysis of these figures consists in the fact that the overall dispersion of the objective function values appears to decrease when introducing synchronisation constraints. While for the model versions without synchronisation, the % diff. indicators may reach up to 40%, this measure decreases to about 14% for the model versions with exact operations synchronisation. For the route similarity ratio, its overall dispersion decreases slightly, from minimum values of 55% for model versions without synchronisation to minimum values of about 60% for the model versions with exact operations synchronisation.

5.3.3 Solution robustness

To assess whether the obtained solutions can realistically be performed in an uncertain setting, a group of Monte Carlo simulations was performed to estimate the probability of the obtained robust solutions being violated. The adopted methodology for these experiments is similar to the ones performed by Munari et al. (2019); Santos et al. (2020).

For each of the instance-parameter combinations with exact synchronisation that obtained near-optimal solutions, 10,000 random scenarios were generated where the actual travel time deviation of each traversed arc took a random value between its nominal value and its worst-case time deviation. The obtained solution was tested for each one of those scenarios, thus evaluating if the solution would be feasible under those specific deviations.

From these experiments, an empirical probability of an obtained solution violating a problem constraint was estimated by dividing the number of infeasible scenarios by the number of total scenarios. Table 11 summarises the results in tabular form, presenting the average probability of constraint violation along with the PoR, specified by instance group.

Table 11: Average Price of Robustness and Probability of Constraint Violation by instance group

		$\Delta = 0.10$						$\Delta = 0.25$						$\Delta = 0.50$					
		C1	C2	R1	R2	RC1	RC2	C1	C2	R1	R2	RC1	RC2	C1	C2	R1	R2	RC1	RC2
$\Gamma = 0$	PoR	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	Prob Violation	0.5%	0.0%	34.4%	14.4%	45.7%	0.0%	19.4%	0.0%	91.9%	41.7%	86.9%	16.5%	28.7%	0.0%	99.7%	71.9%	98.2%	44.8%
$\Gamma = 1$	PoR	0.0%	0.0%	0.7%	0.0%	0.1%	0.0%	0.2%	0.0%	2.6%	0.0%	0.4%	0.1%	1.8%	0.0%	7.6%	0.8%	3.9%	0.6%
	Prob Violation	0.5%	0.0%	4.0%	0.1%	2.6%	0.0%	0.0%	0.0%	17.2%	0.9%	0.2%	0.1%	8.8%	0.0%	56.0%	8.5%	8.8%	0.0%
$\Gamma = 5$	PoR	0.1%	0.0%	1.3%	0.0%	0.1%	0.0%	0.1%	0.0%	3.3%	0.1%	0.6%	0.4%	4.5%	0.0%	11.3%	1.2%	4.0%	0.6%
	Prob Violation	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
$\Gamma = 10$	PoR	0.1%	0.0%	1.3%	0.0%	0.1%	0.0%	0.2%	0.0%	2.9%	0.1%	0.6%	0.4%	4.5%	0.0%	11.6%	1.1%	4.0%	0.6%
	Prob Violation	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Soyster	PoR	0.1%	0.0%	1.7%	0.3%	0.1%	0.0%	0.2%	0.0%	3.3%	0.3%	0.6%	0.2%	4.5%	0.0%	11.6%	1.3%	4.0%	0.6%
	Prob Violation	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

The results show that deterministic solutions are very likely to violate some problem constraint, with instance groups R1 and RC1 being the most problematic. For normalised deviations of $\Delta \geq 0.25$ and above, the instances from these groups are highly likely to violate some problem constraint if deterministic solutions were adopted, with average probabilities around 90%. On the other hand, the deterministic solutions from instance groups C1 and C2 can better accommodate potential travel time deviations, with average probabilities ranging from 0% for lower deviations ($\Delta = 0.10$) to 29% ($\Delta = 0.50$). Nevertheless, this latter probability is still quite high.

As the budget of uncertainty increases, these probabilities decrease as worse but more robust solutions are adopted. For instance, by considering $\Gamma = 1$, and at the expense of solutions that are, on average, 8% more costly, the probability of instances in group R1 violating a problem constraint is decreased from nearly 100% to 56% (for $\Delta = 0.50$). This analysis can be extended to other instance groups, where the probabilities of constraint violation also decreased significantly.

The results show that considering a Γ value higher than 5 is hardly worthwhile since the average probability values already are 0% for this budget of uncertainty. Nevertheless, choosing higher

budgets appears not to yield significant additional costs since the PoR values tend to stabilise. This analysis is consistent with the conclusions obtained previously, where solutions with these budgets of uncertainty are very similar to the ones obtained by the Soyster models due to the small instance sizes.

In summary, the results suggest that for instances of similar size to the ones tested here, adopting $\Gamma = 5$ is a safe choice for a conservative scenario, since the obtained solutions with this budget can withstand the overwhelming majority of uncertainty that travel times can entail. For a less conservative scenario, at least $\Gamma = 1$ would be recommended, where some risk of constraint violation is accepted. However, this last recommendation does not apply to instance group R1 with $\Delta \geq 0.25$, since the risk of constraint violation is still very high in these cases.

6 Conclusions and future work

The work presented in this paper is one of the first studies of a vehicle routing problem subject to travel time uncertainty and synchronisation constraints. More precisely, we consider a robust optimisation approach based on the budget of uncertainty concept devised by Bertsimas and Sim (2004). This approach, which has been popularised in the literature, allows for finer control of the degree of conservatism that the decision maker may want to adapt in their planning activities.

Although robust optimisation is typically known for being more tractable than alternative approaches, such as stochastic optimisation, modelling and embedding robustness into a mathematical model may not be trivial, depending on the specific source of uncertainty being considered. Considering travel times as the source of uncertainty for this problem raised a few challenges in this regard. The approach proposed by Munari et al. (2019), upon which this paper was built, solved part of the problem, specifically concerning the modelling of arrival time variables, by introducing an additional index into these variables, γ, which represents the number of worst-case travel times that have occurred thus far in a route. However, when considering synchronisation, because the worst-case occurrences are not known ahead of time, this requires the mathematical model to have to introduce synchronisation constraints for variables of all possible γ combinations between routes, thus leading to a substantial increase in the number of constraints involving synchronised tasks.

To solve this robust optimisation problem, three mathematical models were proposed. The general 3-index model, model [RVRPSync3], is a more flexible, although less scalable, formulation. The computational experiments showed that this formulation generates large-scale and hardly tractable models, especially for instances with a significant number of synchronisation constraints. The use case for this model should, therefore, be smaller instances that require the higher degree of flexibility that this formulation enables. On the other hand, the 2-index models should be used when possible, since their scalability is much better than the more flexible model.

The differences in model scalability have an impact on model performance. The 2-index models, namely model [RVRPSync1], consistently outperform more general models due to fewer variables and constraints, although still a significant number of instances could not be solved to optimality.

The proposed solution approach was developed under the branch-and-cut framework of a commercial solver and it was able to improve from a standard solver approach by increasing the number of optimally solved instances and decreasing the average optimality gaps. The tailored approach resorted to several initialisation procedures, valid inequalities and cutting planes were implemented to facilitate convergence and reduce initial solver overhead by removing the most complex constraints from the initial model. However, the obtained results suggest that the use of lazy constraints, where standard problem constraints are removed from the initial formulation and introduced only if a candidate solution violates any of those constraints, has provided more

consistent improvements when compared to a standalone solver approach or an approach resorting to cutting planes tailor-made to a candidate solution found by the solver. Although the devised computational experiments were arguably extensive, it should be emphasised that the obtained results are limited by the fact that these experiments were performed by using instances with 25 customers, whose size can be considered modest. The obtained conclusions may change if larger instances were to be considered. Within the scope of branch-and-cut algorithms, further work could address the development of additional valid inequalities that could further accelerate convergence for the remaining problem instances that were unable to be solved to optimality. However, other approximate solution methods may also be feasible to develop, such as heuristic or matheuristic approaches.

When comparing solutions between instances with different types of synchronisation, it was possible to conclude that less restrictive synchronisation constraints do not necessarily yield different solutions. However, it should be considered that these results may be limited by the assumptions of the instances adopted for these experiments. To provide a more accurate conclusion, further work could possibly consider experiments with synchronisation with larger maximum offsets between synchronised tasks.

It is also relevant to emphasise the underlying limitations of the adopted robust optimisation approach, which considers that each route has its own budget of uncertainty. This approach makes the degree of conservatism of a robust-feasible solution highly dependent on the number of routes used. This eventually leads to the conclusion that two robust-feasible solutions with a different number of routes may be evaluated in terms of their robust-feasbility with a very disparate global number of deviations. Future work may be devoted to devising alternative modelling approaches to consider robust optimisation with a global budget of uncertainty that could be evenly or unevenly distributed among routes.

Acknowledgements

This work is co-financed by Component 5 - Capitalization and Business Innovation, integrated in the Resilience Dimension of the Recovery and Resilience Plan within the scope of the Recovery and Resilience Mechanism (MRR) of the European Union (EU), framed in the Next Generation EU, for the period 2021 - 2026, within project AgendaTransform, with reference 34. The first author also acknowledges the support of the Ph.D. Grant SFRH/BD/136314/2018, awarded by the Portuguese Science and Technology Foundation (FCT) and financed by the European Social Fund (ESF) and by National Funds of the Portuguese Ministry of Science, Technology and Higher Education (MCTES) through the Human Capital Operational Programme (POCH).

References

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., & Requejo, C. (2012). Layered formulation for the robust vehicle routing problem with time windows. In *Lecture notes in computer science* (pp. 249–260). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-32147-4_23

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., & Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers & Operations Research*, *40*(3), 856–866. doi: 10.1016/j.cor.2012.10.002

Anderluh, A., Larsen, R., Hemmelmayr, V. C., & Nolz, P. C. (2019). Impact of travel time uncertainties on the solution cost of a two-echelon vehicle routing problem with synchronization. *Flexible Services and Manufacturing Journal*, *32*(4), 806–828. doi: 10.1007/s10696-019-09351-w

Ascheuer, N., Fischetti, M., & Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, *36*(2), 69–79. doi: 10.1002/1097-0037(200009)36:2<69::aid-net1>3.0.co;2-q

Averbakh, I. (2001). On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, *90*(2), 263–272. doi: 10.1007/pl00011424

Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations Research*, *52*(1), 35–53. doi: 10.1287/opre.1030.0065

Birge, J. R., & Louveaux, F. (2011). Basic properties and theory. In *Introduction to stochastic programming* (pp. 103–161). Springer New York. doi: 10.1007/978-1-4614-0237-4_3

Bredström, D., & Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, *191*(1), 19–31. doi: 10.1016/j.ejor.2007.07.033

Campos, R., Coelho, L. C., & Munari, P. (2024). New formulations for the robust vehicle routing problem with time windows under demand and travel time uncertainty. *OR Spectrum*. doi: 10.1007/s00291-024-00781-z

Carlsson, J. G., & Delage, E. (2013). Robust partitioning for stochastic multivehicle routing. *Operations Research*, *61*(3), 727–744. doi: 10.1287/opre.2013.1160

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, *29*(1), 33–51. doi: 10.1016/s0305-0548(00)00056-3

Coindreau, M.-A., Gallay, O., & Zufferey, N. (2021). Parcel delivery cost minimization with time window constraints using trucks and drones. *Networks*. doi: 10.1002/net.22019

Dohn, A., Rasmussen, M. S., & Larsen, J. (2011). The vehicle routing problem with time windows and temporal dependencies. *Networks*, *58*(4), 273—289. doi: 10.1002/net.20472

Drexl, M. (2012). Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science*, *46*(3), 297–316. doi: 10.1287/trsc.1110.0400

Enzi, M., Parragh, S. N., & Puchinger, J. (2021). The bi-objective multimodal car-sharing problem. *OR Spectrum*, *44*(2), 307–348. doi: 10.1007/s00291-021-00631-2

Froger, A., Jabali, O., Mendoza, J. E., & Laporte, G. (2022). The electric vehicle routing problem with capacitated charging stations. *Transportation Science*, *56*(2), 460–482. doi: 10.1287/trsc.2021.1111

Furian, N., O'Sullivan, M., Walker, C., & Vössner, S. (2018). Evaluating the impact of optimization algorithms for patient transits dispatching using discrete event simulation. *Operations Research for Health Care*, *19*, 134–155. doi: 10.1016/j.orhc.2018.03.008

Goel, A., & Meisel, F. (2013). Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, *231*(1), 210–228. doi: 10.1016/j.ejor.2013.05.021

Gorissen, B. L., Yanıkoğlu, İ., & den Hertog, D. (2015). A practical guide to robust optimization. *Omega*, *53*, 124–137. doi: 10.1016/j.omega.2014.12.006

Gounaris, C. E., Wiesemann, W., & Floudas, C. A. (2013). The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, *61*(3), 677–693. doi: 10.1287/opre.1120.1136

Hashemi Doulabi, H., Pesant, G., & Rousseau, L.-M. (2020). Vehicle routing problems with synchronized visits and stochastic travel and service times: Applications in healthcare. *Trans-*

portation Science, *54*(4), 1053–1072. doi: 10.1287/trsc.2019.0956

Hoogeboom, M., Adulyasak, Y., Dullaert, W., & Jaillet, P. (2021). The robust vehicle routing problem with time window assignments. *Transportation Science*, *55*(2), 395–413. doi: 10 .1287/trsc.2020.1013

Hu, C., Lu, J., Liu, X., & Zhang, G. (2018). Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research*, *94*, 139–153. doi: 10.1016/j.cor.2018.02.006

Hà, M. H., Nguyen, T. D., Nguyen Duy, T., Pham, H. G., Do, T., & Rousseau, L.-M. (2020). A new constraint programming model and a linear programming-based adaptive large neighborhood search for the vehicle routing problem with synchronization constraints. *Computers & Operations Research*, *124*, 105085. doi: 10.1016/j.cor.2020.105085

Lee, C., Lee, K., & Park, S. (2012). Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society*, *63*(9), 1294–1306. doi: 10.1057/jors.2011.136

Li, H., Wang, H., Chen, J., & Bai, M. (2020). Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B: Methodological*, *138*, 179–201. doi: 10.1016/j.trb.2020.05.010

Lu, D., & Gzara, F. (2019). The robust vehicle routing problem with time windows: Solution by branch and price and cut. *European Journal of Operational Research*, *275*(3), 925–938. doi: 10.1016/j.ejor.2018.12.019

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, *100*(2), 423–445. doi: 10.1007/s10107-003-0481-8

Mankowska, D. S., Meisel, F., & Bierwirth, C. (2013). The home health care routing and scheduling problem with interdependent services. *Health Care Management Science*, *17*(1), 15–30. doi: 10.1007/s10729-013-9243-1

Meisel, F., & Kopfer, H. (2012). Synchronized routing of active and passive means of transport. *OR Spectrum*, *36*(2), 297–322. doi: 10.1007/s00291-012-0310-7

Mourad, A., Puchinger, J., & Van Woensel, T. (2020). Integrating autonomous delivery service into a passenger transportation system. *International Journal of Production Research*, *59*(7), 2116–2139. doi: 10.1080/00207543.2020.1746850

Munari, P., Moreno, A., Vega, J. D. L., Alem, D., Gondzio, J., & Morabito, R. (2019). The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, *53*(4), 1043–1066. doi: 10.1287/trsc.2018.0886

Ordóñez, F. (2010). Robust vehicle routing. In *Risk and optimization in an uncertain world* (pp. 153–178). INFORMS. doi: 10.1287/educ.1100.0078

Parragh, S. N., & Doerner, K. F. (2018). Solving routing problems with pairwise synchronization constraints. *Central European Journal of Operations Research*, *26*(2), 443–464. doi: 10.1007/s10100-018-0520-4

Santos, M. J., Curcio, E., Mulati, M. H., Amorim, P., & 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. doi: 10.1016/j.tre.2020.101888

Shi, Y., Zhou, Y., Ye, W., & Zhao, Q. Q. (2020). A relative robust optimization for a vehicle routing problem with time-window and synchronized visits considering greenhouse gas emissions. *Journal of Cleaner Production*, *275*, 124112. doi: 10.1016/j.jclepro.2020.124112

Soares, R., Marques, A., Amorim, P., & Parragh, S. N. (2024). Synchronisation in vehicle rout-

ing: Classification schema, modelling framework and literature review. *European Journal of Operational Research*, *313*(3), 817–840. doi: 10.1016/j.ejor.2023.04.007

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*(2), 254–265. doi: 10.1287/opre.35.2.254

Soyster, A. L. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, *21*(5), 1154–1157. doi: 10.1287/opre.21.5.1154

Sungur, I., Ordóñez, F., & Dessouky, M. (2008). A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, *40*(5), 509–523. doi: 10.1080/07408170701745378

A Formulation for the vehicle routing problem with synchronisation

A deterministic formulation for the VRPSync, model [VRPSync], is presented. The proposed formulation takes advantage of the notation previously defined in the paper.

We consider the following decision variables:

$$x_{ij} \quad \begin{cases} 1 & \text{if arc } (i,j) \in \mathcal{A} \text{ is traversed} \\ 0 & \text{otherwise} \end{cases} \tag{48}$$

$$y_{op} \quad \begin{cases} 1 & \text{if operation } (o,p) \in \mathcal{R} \text{ is performed} \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

$$u_{ij} \quad \text{Load of vehicle when traversing arc } (i,j) \in \mathcal{A} \tag{50}$$

$$t_i \quad \text{Arrival time at task } i \in \mathcal{N}_0 \tag{51}$$

Note: Variables y_{op} may be relaxed to continuous variables.

Model [VRPSync] The VRPSync can be formulated as the following mixed-integer linear program:

$$\text{[VRPSync]} \qquad \min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k \tag{52}$$

subject to

$$e_j \leq \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \leq 1 \qquad \forall j \in \mathcal{N} \tag{53}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{i:(j,i)\in\mathcal{A}} x_{ji} = 0 \qquad \forall j \in \mathcal{N} \tag{54}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j} = \sum_{i:(i,n+1)\in\mathcal{A}} x_{i,n+1} \tag{55}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j} \leq K \tag{56}$$

$$u_{ij} \leq Q x_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{57}$$

$$\sum_{i:(i,j)\in\mathcal{A}} u_{ij} + q_j \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \leq \sum_{i:(i,j)\in\mathcal{A}} u_{ji} \qquad \forall j \in \mathcal{N}_0 \setminus \{0\} \tag{58}$$

$$t_j \leq T \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \qquad \forall j \in \mathcal{N}_0 \tag{59}$$

$$t_i + (s_i + d_{ij})\, x_{ij} \leq t_j + T(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A} \tag{60}$$

$$\sum_{i:(i,j)\in\mathcal{A}} a_j x_{ij} \leq t_j \leq \sum_{i:(i,j)\in\mathcal{A}} b_j x_{ij} \qquad \forall j \in \mathcal{N} \tag{61}$$

$$\sum_{l:(l,o)\in\mathcal{A}} x_{lo} + \sum_{l:(l,p)\in\mathcal{A}} x_{lp} - 1 \leq y_{op} \qquad \forall (o,p) \in \mathcal{R} \tag{62}$$

$$y_{op} \leq \sum_{l:(l,o)\in\mathcal{A}} x_{lo} \qquad \forall (o,p) \in \mathcal{R} \tag{63}$$

$$y_{op} \leq \sum_{l:(l,p)\in\mathcal{A}} x_{lp} \qquad \forall (o,p) \in \mathcal{R} \tag{64}$$

$$e_{op} \leq y_{op} \leq 1 \qquad \forall (o,p) \in \mathcal{R} \tag{65}$$

$$t_o + \lambda_{op} \leq t_p + T\,(1 - y_{op}) \qquad \forall (o,p) \in \mathcal{R} \tag{66}$$

$$t_o + \mu_{op} + T\,(1 - y_{op}) \geq t_p \qquad \forall (o,p) \in \mathcal{R} \tag{67}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{68}$$

$$y_{op} \in \{0,1\} \qquad \forall (o,p) \in \mathcal{R} \tag{69}$$

$$0 \leq t_i \leq T \qquad \forall i \in \mathcal{N}_0 \tag{70}$$

$$0 \leq u_{ij} \leq Q \qquad \forall i \in \mathcal{N}_0 \tag{71}$$

The problem's objective function minimises the total travel costs, as defined in Expression (52).

Constraints (53) ensure that every task is performed at most once, and they force mandatory tasks to be performed; they are *global task constraints*. In this formulation, we assume that no tasks are being performed more than once.

Constraints (54) establish the inflow and outflow conservation: vehicles entering a task node must also exit it. Constraints (55) ensure that every vehicle starts and ends its route at the depot, and constraints (56) bound the number of vehicles used.

Constraints (57) are linking constraints between variables x_{ij} and u_{ij}; they impose that the load of a vehicle when traversing arc (i,j) cannot be different from zero if no vehicle traverses said arc ($x_{ij} = 0 \implies u_{ij} = 0$). Constraints (58) establish the load of a vehicle when entering and leaving a task: the difference between the loads must account for the demand of the task being performed.

Constraints (59) are linking constraints between variables x_{ij} and t_i; they impose that the earliest arrival time of vehicle k at task i cannot be different from zero if the vehicle does not perform said task ($x_{ij} = 0 \implies t_i = 0$). Constraints (60) establish vehicles arrival times to tasks. They also serve as sub-tour elimination constraints and are an alternative to the Miller-Tucker-Zemlin constraints. Constraints (61) establish lower and upper bounds to the earliest arrival time of a vehicle to a given task, according to its desired time windows.

Constraints (62)–(64) are linking constraints that set the values of decision variables y_{op} based on the values of variables x_{ij}. Constraints (62) set variable y_{op} to 1 if tasks o and p of operation $(o,p) \in \mathcal{R}$ are being performed. Constraints (63) and (64) set the opposite case: when one of the tasks is not being performed, then the value of y_{op} is forcefully set to zero.

Constraints (65) ensure that operations are performed at most once, and they force mandatory operations to be performed, similarly to what happens in constraints (53).

Constraints (66) correspond to what Soares et al. (2024) consider to be lower-bounding (LB) synchronisation constraints: they ensure that, for a given synchronised operation (o,p), the earliest start time of task p can only be performed λ_{op} time units after the start time of task o. In other words, these constraints establish a lower-bound to the earliest start time of task p based on the earliest start time of task o. Analogously, constraints (67) correspond to upper-bounding (UB) synchronisation constraints: they state that, for a given synchronised operation (o,p), if o is performed before p, then the earliest start time of task p can only be performed up to μ_{op} time units after the start time of o. In other words, these constraints establish an upper-bound to the earliest start time of task p based on the earliest start time of task o.

Constraints (68)–(71) establish the decision variable domains.

B General three-index formulation for the robust vehicle routing problem with synchronisation

The general 3-index formulation for the Robust VRPSync, model [RVRPSync3], is presented. The proposed formulation takes advantage of the notation previously defined in the paper. Therefore, this section will only present the notation that needs to be redefined for the purpose of this formulation.

Table 12 presents the sets and parameters that are redefined in this mathematical formulation.

Table 12: List of sets and parameters

Sets		
$\mathcal{K}$	Set of vehicles or routes	$\mathcal{K} = \{k_1, k_2, ..., k_K\}$
Parameters		
General parameters:		
c_{ij}^k	Cost for traversing arc $(i,j) \in \mathcal{A}$ by vehicle $k \in \mathcal{K}$	
Q^k	Capacity of vehicle $k \in \mathcal{K}$	
Parameters related with operations:		
e_{op}	takes value 1, if operation $(o,p) \in \mathcal{R}$ is mandatory; 0, otherwise	
v_{op}	takes value 1, if operation $(o,p) \in \mathcal{R}$ can be performed by the same vehicle; 0, otherwise	
w_{op}	takes value 1, if operation $(o,p) \in \mathcal{R}$ can be performed by different vehicles; 0, otherwise	

We consider the following decision variables:

$$x_{ij}^k \quad \begin{cases} 1 & \text{if arc } (i,j) \in \mathcal{A} \text{ is traversed by vehicle } k \in \mathcal{K} \\ 0 & \text{otherwise} \end{cases} \tag{72}$$

$$y_{op}^{kk'} \quad \begin{cases} 1 & \text{if operation } (o,p) \in \mathcal{R} \text{ is performed by routes } k \text{ and } k', \text{ respectively} \\ 0 & \text{otherwise} \end{cases} \tag{73}$$

$$u_{ij}^k \quad \text{Load of vehicle } k \in \mathcal{K} \text{ when traversing arc } (i,j) \in \mathcal{A} \tag{74}$$

$$t_i^{k\gamma} \quad \text{Earliest arrival time of vehicle } k \in \mathcal{K} \text{ at task } i \in \mathcal{N}_0 \text{ when } \gamma \leq \Gamma \tag{75}$$

travel times have so far reached their worst-case values

Note: Variables $y_{op}^{kk'}$ may be relaxed to continuous variables.

Model [RVRPSync3] The Robust VRPSync can be formulated as the following mixed-integer linear program:

$$\text{[RVRPSync3]} \qquad \min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k \tag{76}$$

subject to

$$e_j \leq \sum_{k \in \mathcal{K}} \sum_{i:(i,j) \in \mathcal{A}} x_{ij}^k \leq 1 \qquad \forall j \in \mathcal{N} \tag{77}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{i:(j,i)\in\mathcal{A}} x_{ji}^k = 0 \qquad \forall j \in \mathcal{N}, k \in \mathcal{K} \tag{78}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j}^k = \sum_{i:(i,n+1)\in\mathcal{A}} x_{i,n+1}^k \qquad \forall k \in \mathcal{K} \tag{79}$$

$$\sum_{j:(0,j)\in\mathcal{A}} x_{0j}^k \leq 1 \qquad \forall k \in \mathcal{K} \tag{80}$$

$$u_{ij}^k \leq Q^k x_{ij}^k \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \tag{81}$$

$$\sum_{i:(i,j)\in\mathcal{A}} u_{ij}^k + q_j \sum_{i:(i,j)\in\mathcal{A}} x_{ij}^k \leq \sum_{i:(i,j)\in\mathcal{A}} u_{ji}^k \qquad \forall j \in \mathcal{N}_0 \setminus \{0\}, k \in \mathcal{K} \tag{82}$$

$$t_i^{k\gamma} + (s_i + d_{ij})\, x_{ij}^k \leq t_j^{k\gamma} + T(1 - x_{ij}^k) \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, \gamma = 0, ..., \Gamma \tag{83}$$

$$t_i^{k(\gamma-1)} + \left(s_i + d_{ij} + \hat{d}_{ij}\right) x_{ij}^k \leq t_j^{k\gamma} + T(1 - x_{ij}^k) \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, \gamma = 1, ..., \Gamma \tag{84}$$

$$\sum_{i:(i,j)\in\mathcal{A}} a_j x_{ij}^k \leq t_j^{k\gamma} \leq \sum_{i:(i,j)\in\mathcal{A}} b_j x_{ij}^k \qquad \forall j \in \mathcal{N}, k \in \mathcal{K}, \gamma = 0, ..., \Gamma \tag{85}$$

$$\sum_{l:(l,o)\in\mathcal{A}} x_{lo}^k + \sum_{l:(l,p)\in\mathcal{A}} x_{lp}^{k'} - 1 \leq y_{op}^{kk'} \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} \tag{86}$$

$$y_{op}^{kk'} \leq \sum_{l:(l,o)\in\mathcal{A}} x_{lo}^k \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} \tag{87}$$

$$y_{op}^{kk'} \leq \sum_{l:(l,p)\in\mathcal{A}} x_{lp}^{k'} \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} \tag{88}$$

$$e_{op} \leq \sum_{k\in\mathcal{K}} \sum_{k'\in\mathcal{K}} y_{op}^{kk'} \leq 1 \qquad \forall (o,p) \in \mathcal{R} \tag{89}$$

$$y_{op}^{kk} = 0 \qquad \forall (o,p) \in \mathcal{R}, k \in \mathcal{K} : v_{op} = 0 \tag{90}$$

$$y_{op}^{kk'} = 0 \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} : k \neq k', w_{op} = 0 \tag{91}$$

$$t_o^{k\gamma} + \lambda_{op} \leq t_p^{k'\gamma'} + T\left(1 - y_{op}^{kk'}\right) \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} : k \neq k', w_{op} = 1, \tag{92}$$
$$\gamma, \gamma' \in \mathbb{N}_0 : \gamma + \gamma' = \Gamma$$

$$t_o^{k\gamma} + \mu_{op} + T\left(1 - y_{op}^{kk'}\right) \geq t_p^{k'\gamma'} \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} : k \neq k', w_{op} = 1, \tag{93}$$
$$\gamma, \gamma' \in \mathbb{N}_0 : \gamma + \gamma' = \Gamma$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \tag{94}$$

$$y_{op}^{kk'} \in \{0,1\} \qquad \forall (o,p) \in \mathcal{R}, k, k' \in \mathcal{K} \tag{95}$$

$$0 \leq t_i^{k\gamma} \leq T \qquad \forall i \in \mathcal{N}_0, k \in \mathcal{K}, \gamma = 0, ..., \Gamma \tag{96}$$

$$0 \leq u_{ij}^k \leq Q^k \qquad \forall i \in \mathcal{N}_0, k \in \mathcal{K} \tag{97}$$

Constraints (77) ensure that every task is performed at most once, and they force mandatory tasks to be performed; they are *global task constraints*. In this formulation, we assume that no tasks are being performed more than once.

Constraints (78) establish the inflow and outflow conservation: vehicles entering a task node must also exit it. Constraints (79) and (80) ensure that every vehicle starts and ends its route at the depot.

Constraints (81) are linking constraints between variables x_{ij}^k and u_{ij}^k; they impose that the load of vehicle k when traversing arc (i,j) cannot be different from zero if the vehicle does not traverse said arc $\left(x_{ij}^k = 0 \implies u_{ij}^k = 0\right)$. Constraints (82) establish the load of vehicle k when entering and leaving a task: the difference between the loads must account for the demand of the task being performed.

Constraints (83) and (84) establish vehicles earliest arrival times to tasks. They also serve as

sub-tour elimination constraints and are an alternative to the Miller-Tucker-Zemlin constraints. Constraints (85) establish lower and upper bounds on the earliest arrival time of a vehicle at a given task.

Constraints (86)–(88) are linking constraints that set the values of decision variables $y_{op}^{kk'}$ based on the values of variables x_{ij}^{k}. Constraints (86) set variable $y_{op}^{kk'}$ to 1 if tasks o and p of operation $(o,p) \in \mathcal{R}$ are being performed by its corresponding vehicles k and k'. Constraints (87) and (88) set the opposite case: when one of the tasks is not being performed, then the value of $y_{op}^{kk'}$ is set to zero.

Constraints (89) ensure that operations are performed at most once, and they force mandatory operations to be performed, similarly to what happens in constraints (77). Constraints (90) are applied to operations that cannot be performed by the same vehicle: in these situations, if the operation is performed, then it must be performed by different vehicles. Constraints (91) are applied to operations that cannot be performed by different vehicles: in these situations, if the operation is performed, then it must be performed by the same vehicle.

Constraints (92) ensure that, for a given synchronised operation (o,p), the start time of task p can only be performed λ_{op} time units after the start time of task o. On the other hand, constraints (93) state that, for a given synchronised operation (o,p), if o is performed before p, then task p must start being performed up to μ_{op} time units after o begins to be performed.

Finally, constraints (94)–(97) establish the decision variable domains.