



HAL
open science

A Branch-Cut-And-Price approach for the Two-Echelon Vehicle Routing Problem with Drones

Sylvain Lichau, Ruslan Sadykov, Julien François, Rémy Dupas

► **To cite this version:**

Sylvain Lichau, Ruslan Sadykov, Julien François, Rémy Dupas. A Branch-Cut-And-Price approach for the Two-Echelon Vehicle Routing Problem with Drones. *Computers and Operations Research*, 2025, 173, pp.106869. 10.1016/j.cor.2024.106869 . hal-04584429v2

HAL Id: hal-04584429

<https://hal.science/hal-04584429v2>

Submitted on 7 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - ShareAlike 4.0 International License

A Branch-Cut-And-Price Approach for the Two-Echelon Vehicle Routing Problem with Drones

Sylvain Lichau^{a,b,*}, Ruslan Sadykov^{b,d}, Julien François^c, Rémy Dupas^c

^a*Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, Talence, F-33400, France*

^b*Univ. Bordeaux, CNRS, INRIA, Bordeaux INP, IMB, UMR 5251, Talence, F-33400, France*

^c*Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, Talence, F-33400, France*

^d*Atopima, 16 place Sainte-Eulalie, Bordeaux, F-33000, France*

Abstract

We present a new set-partitioning model for the two-echelon vehicle routing problem with drones (2E-VRP-D). In this model, partial routes corresponding to drone movements are enumerated by an efficient dynamic program. A state-of-the-art exact branch-cut-and-price algorithm solves the model using a labeling algorithm for the pricing problems. We further propose an adaptation of the well-known rounded capacity cuts, specifically tailored to address this problem, as well as pre-processing methods designed to enhance the algorithm's efficiency. In addition, we propose an effective heuristic branch-cut-and-price based on the exact algorithm. Computational experiments demonstrate that the exact algorithm solves all instances from the 2E-VRP-D literature and significantly increases the size of optimally solved clustered instances. Finally, a sensitivity analysis is conducted to assess the impact of the proposed enhancements.

1. Introduction

In recent years, the quick growth of e-commerce and the continuous urbanization of populations have resulted in a significant increase in demand for home delivery services, overloading last-mile logistics. To address this challenge, the integration of drones in delivery operations presents a promising solution. In addition to paralleling deliveries, allowing customers to receive their packages faster, the use of drones would also decrease costs for the provider (Wang, 2015). The pollution emitted by the deliveries would diminish because drones are less polluting than trucks. Moreover, traffic congestion could be alleviated substantially by decreasing the volume of truck-based deliveries in urban areas (Figliozzi, 2017).

Numerous pilot projects involving drone deliveries have been successful, and

*Corresponding author

Email address: sylvain.lichau@u-bordeaux.fr (Sylvain Lichau)

many logistics companies are actively working on integrating drones into their existing delivery systems. One of the first successful implementations of a drone/truck delivery system, to our knowledge, was established in France in 2016, specifically targeting a remote area (DPDgroup, 2016). In 2017, a pilot project named Vans & drones combining these two types of vehicles to test on-demand delivery in urban areas was conducted in Zürich (Roca-Riu and Menendez, 2019). Over a period of ten days, they successfully delivered 50 packages, with delays not exceeding two hours. Since 2017, Swiss Post has permanently deployed drones to transfer blood samples between hospitals in Lugano, Zürich, and Bern (Post, 2023). More recently, Amazon has announced the launch of drone deliveries in Italy, the UK, and the US by the end of 2024 (Amazon, 2023).

Despite the promising impact of drone-assisted delivery and the existence of real-world applications, several challenges arise from its large-scale adoption, particularly regarding the close interaction between this new technology and the citizens. Drone use raises security issues as drones must be completely fool-proof and ready to manage various problems, such as extreme weather, technical malfunctions, breakdowns, signal loss, or potential attacks. Drones are also subject to prohibited air zones and other legal restrictions (Lin et al., 2018). Finally, the public’s risk beliefs, such as privacy invasions or injuring people due to malfunction, are also significant obstacles to large-scale implementations of drone deliveries. (Zhu, 2019). Garg et al. (2023) present a literature review of the potential benefits, challenges, and limitations of using drones in last-mile delivery.

Due to the potential evolution of factors such as public opinion, regulation, and technology, it is particularly important to understand how deliveries with drones will impact cities and how to design them in the best possible way. In line with these future developments, the field of operations research is tackling several variants of drone delivery problems, as reviewed in (Macrina et al., 2020) and (Chung et al., 2020). Researchers are considering different truck-drone cooperation systems. In the classic vehicle routing problem with drones (VRP-D), each truck is paired with a single drone. The truck must be parked for the drone to launch and land. The truck continues its route while the drone delivers packages to nearby customers. Many different cooperation systems exist, leading to a large variety of problems.

We study the Two-Echelon Vehicle Routing Problem with Drones (2E-VRP-D). In this problem, a set of vehicles and autonomous drones must deliver packages to customers from a depot. The objective is to minimize the total duration of the routes. This problem is general enough to apply to various real-world situations. It can be summarized as follows:

- Each truck carries an adjustable number of drones. Each drone and its necessary equipment reduce the associated truck payload capacity. Indeed, the technological gap between implementing a truck that carries one drone and a truck that carries several is relatively small, yet it significantly

improves delivery efficiency.

- Each truck and its drone(s) are fully synchronized, i.e., drones are assigned to a truck. A model where drones could change trucks en route would be complex to implement on a real-world application for security and responsibility reasons. To achieve a compromise between implementation simplicity and delivery efficiency, a drone can only fly when its associated truck is parked at a customer’s location. Furthermore, the drones can operate multiple round trips from the same parking spot. The selection of each vehicle’s number of carried drones is made individually.
- Each drone has a limited capacity and, for security reasons, can carry only one package per flight. Drones are considered autonomous. Before each launch, a setup time is required to load the package and change the battery. Drones, utilizing removable batteries, are assumed to be fully recharged upon leaving the truck. Carrying sufficient batteries for deliveries does not introduce any significant transport constraints. A stationary truck during drone flights allows a quick reaction in case of an incident. The maximum flight distance is constrained by the drone’s battery capacity and the weight of the payload.
- Each customer is characterized by a demand, a deadline, and a service time dependent on the means of transport (i.e., truck or drone). Some customers cannot be served by drone. As drones are considered autonomous, the driver can serve the customer where the truck is parked, while the drones independently make other deliveries from the same customer.

Therefore, the 2E-VRP-D is a highly relevant and pragmatic problem as it is relatively straightforward to implement in the delivery process. It is related to the two-echelon vehicle routing problem (2E-VRP), see (Sluijk et al., 2023), because truck deliveries can be considered as a first echelon, while drone deliveries represent a second echelon. In this context, Zhou et al. (2023) proposed a branch-and-price approach that optimally solves instances of up to 30 customers in less than 3 hours for the 2E-VRP-D. Given that real-world last-mile delivery problems commonly contain a higher number of customers to serve, it is necessary to be able to solve larger instances in a reasonable time. We propose a branch-cut-and-price algorithm for the 2E-VRP-D using recent advances from the Capacitated Vehicle Routing Problem (CVRP). The main contributions of this paper are the following:

- A new branch-cut-and-price approach with state-of-the-art components based on enumerating partial routes corresponding to drone operations in the subproblems. The enumeration is modeled as a scheduling problem and solved using an efficient dynamic program. An adaptation of the rounded capacity cuts strengthens the algorithm cuts to better suit this problem, by considering drones’ weight in the truck. Additionally, we propose preprocessing methods to eliminate dominated and infeasible enumerated partial routes.

- A heuristic branch-cut-and-price based on the restriction of the subproblem solution space. This heuristic demonstrates particular efficiency in instances where customers are located randomly. In such cases, drones launched from a vehicle parked at a customer can serve a large number of nearby customers, creating a large "drone-reachable neighborhood".
- We introduce six new sets of medium-size instances with 40, 100, 120, 130, 140 and 150 customers extending the dataset of small-size instances proposed in (Zhou et al., 2023) and based on (Chen et al., 2021), (Solomon, 1987) and (Gehring and Homberger, 1999). Extensive experiments on the instances show that our methods can solve clustered instances with up to 120 customers, significantly expanding the size of the solved instances compared to the existing literature. We also conduct sensitivity analyses on the adapted rounded capacity cuts and the preprocessing methods.

In section 2, we present a literature review with a primary focus on exact methods for solving vehicle routing problems with drones. In section 3, we formally define the problem being addressed. In section 5, we present the pricing problem and propose our solution approach. In section 4, we present our methodology, detailing the branch-cut-and-price algorithm along with the proposed improvements. In section 6, we present the heuristic branch-cut-and-price. Finally, in section 7, we provide computational experiments highlighting the strength of our method.

2. Literature review

The vehicle routing problem with drones (VRPD) class of problem is a variant of the classic vehicle routing problem (VRP) that integrates drones into the deliveries. The first part of this section covers the variants with drones of the traveling salesman problem (TSP) (i.e., single tour), and the second part covers the variants with drones of the VRP (i.e., multiple tours).

2.1. Drones integrated in TSP

The interest to incorporate drones into the routing problem emerged with the work of Murray and Chu (2015), in which a drone and a truck cooperate to solve a TSP-like problem called the flying sidekick traveling salesman problem (FSTSP). They presented a MIP model and two heuristics to solve instances with up to 10 customers. Various extensions were proposed, as in (Agatz et al., 2018) that allowed trucks to wait for the drones at the nodes and proposed a MILP and two heuristics based on the route-first, cluster-second method to solve the problem. The authors also experimentally analyze the time saving achieved by using a truck paired with a drone versus a truck alone. Bouman et al. (2018) used a dynamic program to solve the TSP-D instances with up to 20 customers. They demonstrated that reducing the number of customers visited by truck during drone flights significantly reduced the computation time without diminishing the solution quality. Yang et al. (2023) consider the robust drone-truck

delivery problem, where a truck and a drone have to serve customers to maximize a profit function under truck route uncertainty. They consider truck travel duration uncertainty is considered due to network congestion. They propose a branch and price algorithm to solve instances with up to 40 customers.

Researchers have studied the cooperation of several drones with the truck to improve efficiency further. In (Karak and Abdelghany, 2019), the authors propose the "Hybrid Vehicle-Drone Routing Problem" (HVDRP), where a single vehicle carries a fleet of drones performing pickup and delivery operations. They propose a MIP formulation along with three heuristics to solve the problem. Murray and Raj (2020) introduced the "Multiple Flying Sidekicks Traveling Salesman Problem" (mFSTSP), which is an extension of the FSTSP with several vehicles. It considers an arbitrary number of heterogeneous drones that can be deployed from the depot or the delivery truck. They provide a MIP formulation solving instances with at most 8 customers and a heuristic approach. Kang and Lee (2021) introduce the Heterogeneous Drone-Truck Routing Problem (HDTRP), where drones with different speeds and batteries are set up on a truck and can fly only when the truck is parked at a customer. They propose a MILP formulation and a branch-and-cut algorithm to solve instances with up to 50 customers optimally.

2.2. Drones integrated in VRP

The use of drones with a set of trucks was first introduced in (Wang et al., 2017), which presents the Vehicle Routing Problem with Drones (VRPD), where multiple trucks are equipped with drones to serve a set of customers. In this problem, the drones carried by a truck can deliver one parcel per flight, and the truck can move between the launch and the retrieval of its drones. The authors analyze different worst-case scenarios and provide upper bounds on the savings induced by using drones. The work is extended in (Wang and Sheu, 2019), but the drones can operate multiple deliveries per trip, and the drone-truck assignment is open. The authors propose a branch-and-price algorithm to solve instances with up to 13 customers. They also provide a sensitivity analysis of the drones' maximal authorized flying duration. Poikonen et al. (2017) consider the battery of the drones, a different metric for the drones and the trucks, as well as a cost objective function. They provide theoretical results regarding the VRPD.

Di Puglia Pugliese and Guerriero (2017) present the first MIP model for an extension of the VRP-D called the vehicle-drone routing problem with time windows (VDRPTW), which also considers multiple drones per truck, time windows, and truck capacity. They also provide a numerical study comparing VRP to VRP-D. They concluded that drones are only advantageous when the drone's transportation cost is lower than the truck's. This work is extended in (Di Puglia Pugliese et al., 2020) where the authors present a MIP model solving up to 15 customer instances. They also compare VRP, VRP-D, and routing problem with drones (RPD), where the drones serve customers directly from the depot, on their CO₂ emission and traffic congestion. Sacramento et al. (2019) address an extension of the VRP-D where the drones can return to the depot

directly. They present an Adaptive Large Neighborhood Search (ALNS) and a MIP model. In (Schermer et al., 2019), the authors consider the Vehicle Routing Problem and En Route Operation (VRPDERO), where a drone can be launched and retrieved on a moving truck. They propose a MIP model and a heuristic. In (Kitjacharoenchai et al., 2020), the objective function is to minimize the total completion time. A truck can only launch or retrieve a single drone upon reaching a node. A drone can deliver parcels to multiple customers before returning to its assigned truck. The authors propose a mixed integer non-linear program (MINLP) that optimally solves instances with up to 9 customers. Tamke and Buscher (2021) proposed a branch-and-cut algorithm for the VRP-D without truck capacity solving up to 30 customer instances. The work is extended in (Tamke and Buscher, 2023), where the speed at which a drone performs a flight must be selected from a discrete set and influence the energy consumption. The authors propose a MIP model, valid inequalities, and preprocessing methods to accelerate the solving. They solve instances with 20 customers optimally and find feasible solutions on 50 customer instances with a heuristic version of their MIP.

Although the VRPD and its extensions are the most studied, other works related to the integration of drones in vehicle routing problems are also considered. Ulmer and Thomas (2018) consider a dynamic variant, the Same-Day Delivery Routing Problem with Heterogeneous Fleets of drones and vehicles (SDDPHF), where customers requests arrive while the vehicles are already on the road. Drones make deliveries from the depot, serving a single customer before coming back to the depot. The authors present a policy function approximation to decide whether a request is fulfilled by a drone or a vehicle. Kitjacharoenchai et al. (2019) introduce the multiple traveling salesman problem with drones (mTSPD), where multiple vehicles and drones serve a set of customers from a depot. They do not limit the number of drones on a vehicle, and the truck-drone assignment is open (i.e., the launch and retrieve operations can be operated by different trucks). The authors propose a MIP model solving instances with up to 9 customers and a heuristic. Dukkanci et al. (2021) propose the Energy Minimizing and Range Constrained Drone Delivery Problem (ERDDP), where trucks move from the depot to parking spots. Drones can fly from a parking spot or from the depot to deliver a single customer. The objective function minimizes a cost function over the energy spent and the operational costs. The drone’s speed directly impacts energy consumption and is considered a decision variable. The authors conduct a parametric analysis of the number of vehicles used. Meng et al. (2023) introduce the multi-visit drone routing problem for pickup and delivery services (MDRP-PD). They propose a MIP and a two-stage heuristic based on simulated annealing. Zhou et al. (2023) present the two-echelon vehicle routing problem with drones (2E-VRP-D). We recall that in this problem, the number of drones on a truck is a decision variable, which allows a better use of the drone fleet. The drones can only fly when their assigned truck is parked at a nearby client. To solve the problem, they presented a MIP, a branch and price, and a heuristic. They managed to solve up to 30 clients to optimality. A similar problem is studied in (Yin et al., 2024), which considers routing

vehicle-and-drone cooperatives to deliver blood products to hospitals. The primary difference between the two problems lies in the objective function, as Yin et al. (2024) penalize the weighted, squared arrival time at each hospital. They use a Benders-decomposition column-generation algorithm to solve instances of up to 45 customers.

To give an overview of the literature regarding exact approaches for drones integrated in VRP, we present a summary table. Let us first recall the notation for the vehicle routing problems with drones introduced in (Tamke and Buscher, 2021). The criteria are the following: number of trucks ($\#t$), number of drones per truck ($\#d$), drone-truck assignment (dta) i.e., whether the drone has an assigned truck or not, launch and retrieve operations (lro), drone range (ran), drone capacity (cap), and objective function (obj). Most criteria are self-explanatory. In Table 1, the keyword 'return' refers to the ability of a truck to return to an already visited customer. The keyword 'loops' refers to the ability of a drone to operate loop operations, in which the start and end locations of the flight are the same.

Criterion	Value	Description	Criterion	Value	Description
$\#t$	1	single truck	ran	t	time
	m	multiple trucks		d	distance
$\#d$	1	single drone	cap	n	number of nodes
	m	multiple drones		e	energy consumption
	u	unlimited number		u	unrestricted
dta	f	fixed	m	1	single customer
	o	open		m	multiple customers
lro	a	return and loops allowed			
	p	return and loops prohibited			
	l	only loops allowed			
obj	min ct	minimize completion time			
	min tc	minimize total costs			
	min wt	minimize customer waiting times			
	bi-obj	minimize total costs and completion time			
	min max ct	minimize maximum completion time			
	min total ct	minimize total completion time			

Table 1: Classification criteria for delivery systems with truck-drone cooperation. (Tamke and Buscher, 2021)

Using the notation defined in Table 1, we present Table 2, providing information about the problem studied, the exact resolutions method and the results of different paper considering the integration of drones to vehicle routing problems. The algorithms are abbreviated as follows: BP stands for branch-and-price, BC for branch-and-cut and BCP for branch-and-cut-and-price.

Reference	Problem	<#t	#d	dta	lro	ran	cap	obj	Approach	Size	Note
Sacramento et al. (2019)	VRP-D	<m	1	f	p	t	1	min tc	MIP	12	
Sacramento et al. (2019)	VRP-D	<m	1	f	p	t	1	min tc	MIP	12	
Di Puglia Pugliese and Guerriero (2017)	VDRPTW	<m	m	f	p	d	1	min tc	MIP	10	Time Window
Meng et al. (2023)	MDRP-PD	<m	1	f	p	e	m	min tc	MIP	11	Pickup and Delivery
Wang and Sheu (2019)	VRPD	<m	m	o	p	t	m	min tc	BP	13	Docking Hubs
Schermer et al. (2019)	VRPDERO	<m	m	f	p	d	1	min max ct	MIP	10	En-route Operation
Kitjacharoenchai et al. (2019)	mTSPD	<m	u	o	p	u	1	min max ct	MIP	9	
Kitjacharoenchai et al. (2020)	VRPD	<m	m	f	p	d	m	min total ct	MINLP	9	
Di Puglia Pugliese et al. (2020)	VRP-D	<m	m	f	p	d	1	min tc	MIP	15	Time Window
Dukkanci et al. (2021)	ERDDP	<m	m	f	l	e	1	min tc	SOCP	37	Drone Speed Variable
Tamke and Buscher (2021)	VRPD	<m	m	f	p	d	1	min max ct	BC	20	
Zhou et al. (2023)	2E-VRP-D	<m	m	f	l	e	1	min total ct	BP	30	Variable nb Drone per Truck
Tamke and Buscher (2023)	VRPD-DSS	<m	m	f	p	e	1	min tc	MIP	20	Drone Speed Variable
Yin et al. (2024)	BPDRP	<m	m	f	l	t	1	min tc	BP	45	Quadratic objective
Our	2E-VRP-D	<m	m	f	l	e	1	min total ct	BCP	120	Variable Drone per Truck

Table 2: Overview of exact approaches for drones integrated in VRP

3. Formulation

We now introduce the addressed problem. The 2E-VRP-D was first defined in (Zhou et al., 2023). In this problem, k_v vehicles and k_d drones have to deliver a set of customers $V = \{1, \dots, n\}$ from a depot denoted 0. Each truck can carry at most D drones.

The following assumptions are made:

- (a) Both drones and vehicles deliver packages, but some customers have to be served by truck.
- (b) A drone is limited by its capacity and battery but is fully recharged upon reaching the vehicle.
- (c) A drone can only carry one package per flight and is synchronized with a single assigned vehicle.
- (d) A drone can only fly when its vehicle is parked at a customer but can take off multiple times from the same customer's location.
- (e) Each drone and its necessary equipment have a weight q_1^d .
- (f) The total capacity of a vehicle, denoted by Q , limits the weight of packages and drones it can carry.
- (g) The service time at a customer delivered by a vehicle and the drone deliveries from this customer occur simultaneously.

We define \mathcal{N}_i , $i \in V$, the drone-reachable neighborhood of i , as the set of customers that can be served by drone from i . For completeness, a formal definition of \mathcal{N}_i , $i \in V$ can be found in Appendix A. We define a drone schedule as an ordered set of round trips for each drone originating from a truck parked at a customer location. The problem is defined by the complete symmetric directed graph $G = (V_+, A)$, where $V_+ = \{0\} \cup V$ is the set of vertices and $A = \{(i, j) \mid i, j \in V_+, i \neq j\}$ is the set of arcs. A customer $i \in V$ is defined by a demand q_i , a vehicle service time s_i^v , a drone service time s_i^d , an autonomous drone setup time o_i , and a deadline \bar{d}_i , which is the latest date a vehicle can arrive to serve the customer i . Travelling time between two nodes $i, j \in V_+$ is noted $c_{i,j}^v$ for a vehicle and $c_{i,j}^d$ for a drone.

We now present our mathematical formulation of the problem. We define a route as the combination of a truck route for the first echelon and a set of drone schedules for the second echelon. Let \mathcal{R}^d , $d \in \{0, \dots, D\}$, be the set of all feasible routes of a vehicle carrying d drones. For every $d \in \{0, \dots, D\}$, every $i \in V$, and every subset $K \subseteq \mathcal{N}_i$, let S_{iK}^d be the set of all optimal drone schedules serving the set of customers K with d drones from the customer i . Set S_{iK}^d may contain several schedules because the arrival date of the truck at customer i may vary, and the customers have deadlines. Hence, the optimal solution may not be feasible for a given truck arrival date. Note that set S_{iK}^d contains the Pareto front, minimizing the maximal drone schedule duration and maximizing the maximal truck arrival date. We note m_s , $s \in S_{iK}^d$ the makespan of the drone schedule s . Let $b_{i,j}^r$ be equal to one if and only if the vehicle moves from i to j in route $r \in \mathcal{R}^d$. Let a_s^r be equal to one if and only if the drones follow drone schedule $s \in S_{iK}^d$ when the vehicle is parked at customer $i \in V$ in route r . Let variable

λ_r , $r \in \mathcal{R}^d$, $d \in \{0, \dots, D\}$ be equal to 1 if the route r is used in the solution and 0 otherwise. We also define the following additional variables :

x_{ij}^d — an integer variable equal to the number of times a vehicle with $d \in \{0, \dots, D\}$ drones travels between node $i \in V_+$ and node $j \in V_+$, $i < j$, regardless of the order of visit.

y^d — a continuous variable equal to the total truck immobility time in routes of vehicles with d drones.

z_j^d — a binary variable equal to one if and only if a vehicle with d drones sends a drone to visit customer $j \in V$.

u^d — the integer variable equal to the number of used vehicles with d drones.

Variables with superscript d belong to subproblem $d \in \{0, \dots, D\}$.

The set partitioning formulation is presented in two parts, using the previous notations. The first part (i.e., equations (2) to (4)) uses the additional variables to model the problem. The second part (i.e., equations (5) to (8)) maps the variable λ_r to the additional variables.

$$\min \sum_{d \in \{0, \dots, D\}} \left(\sum_{i \in V_+} \sum_{j \in V_+} c_{i,j}^v \times x_{ij}^d + y^d \right) \quad (1)$$

$$\text{s.t.} \sum_{d \in \{0, \dots, D\}} \left(\sum_{j=1}^{i-1} \frac{1}{2} x_{ji}^d + \sum_{j=i+1}^n \frac{1}{2} x_{ij}^d + z_i^d \right) = 1, \quad \forall i \in V, \quad (2)$$

$$\sum_{d \in \{0, \dots, D\}} u^d \leq k_v, \quad (3)$$

$$\sum_{d \in \{0, \dots, D\}} d \times u^d \leq k_d, \quad (4)$$

$$x_{ij}^d = \sum_{r \in \mathcal{R}^d} (b_{ij}^r + b_{ji}^r) \times \lambda_r, \quad \forall d \in \{0, \dots, D\}, i, j \in V_+, i < j, \quad (5)$$

$$y^d = \sum_{r \in \mathcal{R}^d} \sum_{j \in V} \sum_{K \in \mathcal{K}_j} \sum_{s \in S_{jK}^d} (m_s \times a_s^r) \times \lambda_r, \quad \forall d \in \{0, \dots, D\}, \forall i \in V, \quad (6)$$

$$z_i^d = \sum_{r \in \mathcal{R}^d} \sum_{j \in V} \sum_{K \in \mathcal{K}_j} \sum_{\substack{s \in S_{jK}^d \\ i \in K}} a_s^r \times \lambda_r, \quad \forall d \in \{0, \dots, D\}, \forall i \in V \quad (7)$$

$$u^d = \sum_{r \in \mathcal{R}^d} \sum_{j \in V} b_{0j}^r \times \lambda_r, \quad \forall d \in \{0, \dots, D\}, \quad (8)$$

$$\lambda_r \in \{0, 1\}, \quad \forall d \in \{0, \dots, D\}, \forall r \in \mathcal{R}^d. \quad (9)$$

The objective function (1) and constraints (2)–(4) belong to the master. The objective function (1) minimizes the total duration of the routes. The constraints (2) ensure that each customer is visited exactly once. The constraints (3) and (4) verify that the solution uses a feasible number of trucks and drones. The constraints (5)–(8) define the mapping between the additional variables and the route variables. Note that additional variables are used mainly for the clarity of the formulation and can be eliminated from the formulation using relations (5)–(8).

4. Branch-cut-and-price algorithm

Because it would be impractical to enumerate the set $\mathcal{R}^d, d \in \{0, \dots, D\}$ due to its exponential size over the number of customers, we solve this formulation using a column and cut generation approach. We use a similar branch-and-cut-and-price (BCP) as in (Pessoa et al., 2020), with the following features. The BCP algorithm iteratively generates new variables (columns) to improve the linear programming (LP) relaxation of an integer program. Based on the LP solution, branching decisions are made, and cutting planes are added to strengthen the relaxation. Two primal heuristics are used in branch and bounds nodes. The first heuristic solves the restricted master problem using a MIP solver and a time limit. The second heuristic is called the diving heuristic, where columns with the most significant fractional values are iteratively set to one, and the remaining problem is solved using column generation.

4.1. Column generation

We use three-stage column generation. The pricing is solved using heuristic labeling algorithms in the two first stages as in (Sadykov et al., 2021), and using the exact labeling dynamic algorithm as in (Pessoa et al., 2020) and (Sadykov et al., 2021). The first two stages generate at most 30 columns, and the last stage at most 150. Automatic dual price smoothing stabilization is used, as in (Pessoa et al., 2018), to help the convergence of the process. The bucket-arc elimination procedure is also used, removing arcs that have proven not to belong to any optimal solution. After each call, we use a route enumeration technique similar to (Baldacci et al., 2008), which tries to enumerate all improving routes for a subproblem. If it succeeds, the future pricing problems will be solved by inspection of the enumerated routes. Furthermore, if the number of enumerated routes in all subproblems is less than a certain threshold (we use 5000 as the threshold), all the routes are added to the restricted master problem and solved by the MIP solver.

4.2. Cutting planes

4.2.1. Adapted rounded capacity cuts

The rounded capacity cuts were first introduced in (Laporte and Nobert, 1983) for the constrained vehicle routing problem (CVRP). Let $C \subset V$ and let h_r be

the number of times $r \in \mathcal{R}^d$ visits C i.e.

$$h_r = \frac{1}{2} \times \sum_{i \in V_+ \setminus C} \sum_{j \in C} b_{\min(i,j), \max(i,j)}^r + \sum_{i \in V_+ \setminus C} \sum_{\substack{K \in \mathcal{N}_i: \\ K \cap C \neq \emptyset}} \sum_{s \in S_{iK}^d} a_s^r \quad \forall d \in D, r \in \mathcal{R}^d \quad (10)$$

These valid inequalities can be formulated as follows:

$$\sum_{d \in D} \sum_{r \in \mathcal{R}^d} \lambda_r \times h_r \geq \left\lceil \frac{\sum_{i \in C} q_i}{Q} \right\rceil, \quad C \subset V \quad (11)$$

However, in this problem, drones occupy a significant portion of the truck's capacity (about 20% per drone in the instances), which must be taken into consideration. Let $k = \left\lfloor \frac{Q}{q_1^d} \right\rfloor$, and we adapt the rounded capacity cuts as follows:

$$\sum_{d \in D} \sum_{r \in \mathcal{R}^d} \lambda_r \times h_r \times (k - d) \geq \left\lceil \frac{\sum_{i \in C} q_i}{q_1^d} \right\rceil, \quad C \subset V \quad (12)$$

We will refer to these as adapted rounded capacity cuts (ARCC). We use the greedy construction heuristic from (Lysgaard et al., 2004) to separate the ARCC.

Algorithm 1 Separation for the adapted rounded capacity cuts

Require: $t \in \mathbb{R}$

```

1: for  $i \in V$  do
2:    $C \leftarrow \{i\}$ 
3:   while  $C \neq V$  do
4:      $C \leftarrow C \cup \{\arg \max_{j \in V \setminus C} (violation(C \cup \{j\}))\}$ 
5:     if  $violation(C) > t$  then
6:        $addCut(C)$ 

```

We describe the separation algorithm in Algorithm (1), where t is a user-defined threshold, the function $violation(C)$ calculates the difference between the right-hand side and the left-hand side in constraints (12) for set C and $addCut(C)$ add the valid inequality to the problem. The algorithm can be explained as follows: starting with a single customer in the set, while not all customers are in the set, we add a customer that is not already in the set, and that maximizes the violation. If the violation of the set exceeds the threshold, we add the corresponding cut to the linear program. We generate at most 100 cuts per round.

4.2.2. Limited memory rank-one cuts

We use limited memory rank-one cuts (lm-R1Cs) similar to (Pecin et al., 2017a), (Pessoa et al., 2020). The R1Cs uses a Chvátal-Gomory rounding on a weighted sum of a subset of constraints (2). When the coefficients are all equal, these cuts are called Subset-Row cuts (Jepsen et al., 2008).

The R1Cs are defined as follows. Let $C \subset V_+$, with multipliers $0 < \rho_i < 1 \quad \forall i \in C$ and let h_i^r be the number of time the vertex i is visited in route r :

$$h_i^r = \frac{1}{2} \sum_{j \in C} b_{\min(i,j), \max(i,j)}^r + \sum_{j \in V_+ \setminus C} \sum_{\substack{K \in \mathcal{K}_j \\ i \in K}} \sum_{s \in S_{jK}^d} a_s^r \quad \forall d \in D, r \in \mathcal{R}^d \quad (13)$$

The following equation is a valid inequality:

$$\sum_{d \in D} \sum_{r \in \mathcal{R}^d} \left[\rho_i \sum_{i \in C} h_i^r \right] \lambda_r \leq \left[\sum_{i \in C} \rho_i \right] \quad (14)$$

Because the R1Cs are non-robust constraints, we use lm-R1Cs instead, where the set C is paired with a memory set M such that $C \subseteq M \subseteq V_+$. $\left[\rho_i \sum_{i \in C} h_i^r \right]$ in equation (14) is replaced by $\alpha(C, M, \rho, P)$ and is calculated during the pricing computation as in algorithm 2.

Algorithm 2 $\alpha(C, M, \rho, P = (v_0 = 0, v_1, \dots, v_k, v_k + 1 = 0'))$

```

1:  $\alpha \leftarrow 0, s \leftarrow 0$ 
2: for  $i = 1$  to  $k$  do
3:   if  $(v_{i-1}, v_i) \notin M$  then
4:      $s \leftarrow 0$ 
5:   if  $v_i \in C$  then
6:      $s \leftarrow s + \rho_{v_i}$ 
7:     if  $s \geq 1$  then
8:        $s \leftarrow s - 1, \alpha \leftarrow \alpha + 1$ 
9: return  $\alpha$ 

```

If the partial path P leaves the set M , all previous visits are forgotten, leading to an easier-to-compute but smaller coefficient. Note that if $M = V$, lm-R1Cs are equivalent to R1Cs. To separate those cuts, we find a violated constraint and calculate its minimal memory set such that the limited memory R1Cs have the same violation.

4.3. Branching

We select a collection of branching candidates from the total number of drones ($\sum_{d \in \{0, \dots, D\}} d \times u^d$), the total number of trucks ($\sum_{d \in \{0, \dots, D\}} u^d$), the number of trucks with d drones (variables u) and the truck travel variables (variables x).

We use a multi-phase strong branching procedure to choose the most promising branching candidate, similar to (Pecin et al., 2017a). The idea is to spend

more time evaluating branching variables in the lowest depth of the branch-and-bound tree where each selection has a greater impact on the overall time and spend less time as the depth increases, taking advantage of the history of previous evaluations.

In the first phase, half the candidates are chosen from history from previous calls of the strong branching procedure. The remaining half is selected in a balanced manner from all branching strategies. The selection method is based on the distance between the variable’s value and the nearest integer. The candidates with the largest distances are selected.

In the second phase, selected candidates from the first phase are evaluated by solving the current restricted master linear program modified for each candidate without column generation. We select the candidates using the product rule (Achterberg, 2007).

In the third phase, selected candidates from the second phase are evaluated by solving the restricted master linear program modified for each candidate, with heuristic column generation but without cut generation. We select the best candidate using the product rule.

5. Pricing subproblem

We design the pricing subproblems as a Resource Constrained Shortest Path Problem (RCSP), where a feasible solution is a feasible route for the 2E-VRP-D. The objective of this problem is to determine a shortest path between a source and a sink in a graph where each arc consumes resources and requires that the remaining resources lie between lower and upper limits.

We solve the subproblems using the bucket graph labeling algorithm presented in (Sadykov et al., 2021). The algorithm tries to extend partial paths called labels to form complete paths. Given their accumulated resources, labels are stored and extended according to so-called buckets, helping greatly with dominance checks. This algorithm supports ng-paths relaxation (Baldacci et al., 2011) as well as the presence of limited rank-1 cuts (Pecin et al., 2017b).

We model a RCSP problem for each $d \in \{0, \dots, D\}$ in a graph defined as follows. Let $\mathcal{G}^d = (\mathcal{V}^d, \mathcal{A}^d)$ be a directed graph, where $\mathcal{V}^d = V_+ \cup V'_+$, with V'_+ a copy of V_+ . Set $\mathcal{A}^d = A^{dT} \cup A^{dD}$ with A^{dT} the set of arcs representing a truck delivery possible with d drones and A^{dD} the set of arcs representing possible drone deliveries with d drones. The source of the graph is node $0'$, and the sink is node 0 . Arcs in A^{dD} go from a vertex $i \in V$ to the vertex $i' \in V'$, which is the copy of i . Let S_{iK}^d be the set of drone schedules with d drone, serving the set of customers $K \subseteq \mathcal{N}_i$ from i . For each $i \in V$, each possible subset $K \subseteq \mathcal{N}_i$, and each drone schedules $s \in S_{iK}^d$, we define an arc in set A^{dD} , labelled $(i, i')^s$. Let a_s^r be equal to one if and only if route $r \in \mathcal{R}^d$ follows this arc, i.e., the drones follow drone schedule $s \in S_{iK}^d$ in route r when the vehicle is positioned at customer $i \in V$. Arcs in A^{dT} go from a vertex $i' \in V'_+$ to a vertex $j \in V^+$. Let $b_{i,j}^r$ be equal to one

if and only if route $r \in \mathcal{R}^d$ follows arc (i', j) . Note that the prime is omitted in the notation without ambiguity, as there is no arc between i and j with $i, j \in V_+$.

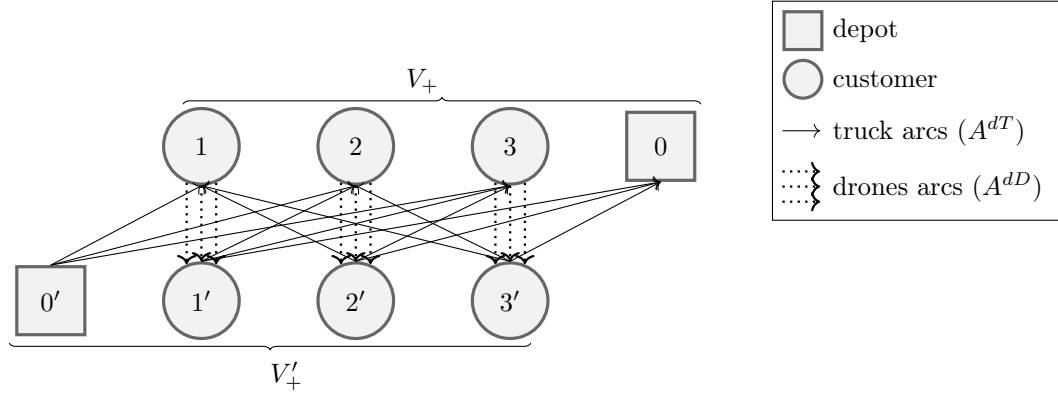


Figure 1: Example of a subproblem graph \mathcal{G}^d

Figure (1) shows an example of a subproblem graph. We represented only three arcs from $i \in V$ to $i' \in V'$ for clarity, but there should be an arc for all $s \in S_{iK}^d, \forall K \subseteq \mathcal{N}_i$.

5.1. Resources for the pricing subproblem

In our model, we use 2 resources, the duration β_d of a route and the capacity β_c of a truck, and we define a pricing subproblem for each $d \in \{0, \dots, D\}$. The objective function is the minimization of the reduced cost. The reduced cost of a route r with a cost c_r is given by: $\bar{c}_r = c_r - \sum_{i \in V} \alpha_{ir} \times \pi_i - \lambda_0^v - d_r \times \lambda_0^d$ where α_{ir} is equal to one if and only if customer i is visited in the route r , d_r the number of drones used in route r , π_i are the dual variables associated with constraints (2), λ_0^v is the dual variable associated with constraint (3) and λ_0^d is dual variable associated with constraint (4).

Let $[l_{a,r}, u_{a,r}]$ be the resource feasibility interval, for each $a \in \mathcal{A}^d$ and $r \in \{\beta_c, \beta_d\}$, i.e. the accumulated resource r has to lie within the interval $[l_{a,r}, u_{a,r}]$ to travel through arc a . Let $q_{a,r}$ be the consumption of resource r on arc a . For the resource capacity, we set:

$$l_{a,\beta_c} = 0 \quad \forall a \in \mathcal{A}^d \quad (15)$$

$$u_{a,\beta_c} = Q - d \times q_1^d \quad \forall a \in \mathcal{A}^d \quad (16)$$

$$q_{a,\beta_c} = \begin{cases} 0 & \forall a = (i, 0), i \in V' \\ q_j & \forall a = (i, j), i \in V'_+, j \in V \\ \sum_{k \in K} q_k & \forall a = (i, i')^s, i \in V, i' \in V', K \in \mathcal{N}_i, s \in S_{iK}^d \end{cases} \quad (17)$$

The truck starts with a remaining capacity of $Q - d \times q_1^d$. The remaining capacity of a truck have to lie between 0 and $Q - d \times q_1^d$ on every arc of the graph. The remaining capacity consumption of an arc is set to the sum of the demands of the customers served, or 0 if the destination is the depot.

Let Δ_s be the maximal starting time of a solution s that will be formally defined in section (5.2.1). For the resource duration, we set:

$$l_{a,\beta_d} = 0 \quad \forall a \in \mathcal{A}^d \quad (18)$$

$$u_{a,\beta_d} = \begin{cases} \bar{d}_j & \forall a = (i, j), i \in V'_+, j \in V_+ \\ \Delta_s & \forall a = (i, i')^s, i \in V, i' \in V', K \in \mathcal{N}_i, s \in S_{iK}^d \end{cases} \quad (19)$$

$$q_{a,\beta_d} = \begin{cases} c_{i,j}^v & \forall a = (i, j), i \in V'_+, j \in V_+ \\ \max(s_i^v, m_s) & \forall a = (i, i')^s, i \in V, i' \in V', K \in \mathcal{N}_i, s \in S_{iK}^d \end{cases} \quad (20)$$

The truck starts at time 0. A truck arc serving a customer can only be used if the accumulated duration lies between 0 and the customer's deadline. A drone arc serving a set of customers can only be used if the accumulated duration lies between 0 and the drone schedule's maximum starting time. The duration consumption of a drone arc is the maximum between the truck's service time at customer i and the duration of the drone schedule, as they are operated in parallel.

5.2. Drone schedules enumeration

The objective is to enumerate all optimal drone schedules from all customers $i \in V$, for all number of drones on the truck $d \in \{1, \dots, D\}$, and for all sets of available customers, $K \subseteq \mathcal{N}_i$. Note that the number of subsets of \mathcal{N}_i is exponential. Because of customers' deadlines, the optimal drone schedule serving the set of customers K depends on the truck arrival time at i . We note $T_i = [c_{0,i}^v, \bar{d}_i]$ the interval containing all possible truck arrival times at customer i . Note that our approach does not require time discretization.

5.2.1. Scheduling problem

Each drone can only deliver a single customer per round-trip and is automatically fully recharged when reaching the truck. Consequently, we can model the drone routing problem for a truck with d drones arriving at customer i at time $t_a \in T_i$ and serving set $K \subseteq \mathcal{N}_i$ customers as a scheduling problem with d parallel identical machines, $|K|$ jobs with deadlines and minimizing the makespan. The arrival time t_a can be treated as a common release date for all jobs and eliminated by subtracting t_a from each job's deadline. This operation is included in the job deadline formula 22. Therefore, the notation of this scheduling problem is $P|\bar{d}_j|C_{\max}$ (Graham et al., 1979). In this problem, waiting times between jobs can be eliminated without compromising the quality of a solution. Therefore, solutions involving waiting time can be omitted.

We model the problem as follows:

Let J_K^i be the set of jobs where each job corresponds to the delivery of a customer in $K \subseteq \mathcal{N}_i$ by drone from $i \in V$. Each job corresponds to a customer's delivery. The processing time of the job is defined by the period of drone unavailability caused by the delivery, which includes the setup time, travel times, and service time. The job's deadline is derived from the customer's delivery deadline, though it is adjusted to align with the definition of processing time. For $t \in J_K^i$, let:

- p_t be its processing time
- \bar{d}_t be its deadline
- v_t be the customer whose delivery it represents.

Formally, we have the following equations:

$$p_t = o_{v_t} + c_{i,v_t}^d + s_{v_t}^d + c_{v_t,i}^d \quad (21)$$

$$\bar{d}_t = \bar{d}_{v_t} + s_{v_t}^d + c_{v_t,i}^d - t_a \quad (22)$$

To solve this problem, a distinction must be made depending on the number of machines.

- When considering only one machine, this scheduling problem can be solved in polynomial time using the Jackson's rule, also known as the Earliest Deadline First (Jackson, 1955), (Stankovic et al., 1998). Therefore, processing the jobs ordered by increasing deadlines yields an optimal solution. Modifying t_a does not affect the sequence of jobs sorted by increasing deadlines. Consequently, for a given t_a , either this solution is optimal or no feasible solution exists.
- When considering more than one machine, $P||C_{\max}$ is known to be NP-hard Błażewicz et al. (2007), therefore, $P|\bar{d}_j|C_{\max}$ is NP-hard.

5.2.2. Dynamic program computing Pareto front

As mentioned earlier, the set of schedules for a given $i \in V, d \in \{0, \dots, D\}, K \subseteq \mathcal{N}_i$ is noted S_{iK}^d . To be more precise, S_{iK}^d is the Pareto front minimizing the

makespan and maximizing the maximal starting time.

This section presents a dynamic program that solves for the Pareto front corresponding to a set S_{iK}^d .

For $s \in S_{iK}^d$, let:

- m_s be its makespan.
- C_t^s be the completion time of job $t \in J_K^i$.
- Δ_s be its maximal starting time, i.e., $\Delta_s = \min_{t \in J_K^i} (\bar{d}_t - C_t^s)$.

The algorithm computes S_{iK}^1 for all $i \in V$ and for all $K \in \mathcal{N}_i$. As explained in the previous section, when using a single machine, we only have to consider the solution given by Jackson's rule. Consequently, solving the problem solely for truck arrival time $t_a = c_{0,i}^v$, the earliest time a truck can arrive at customer i , is sufficient to obtain S_{iK}^1 . Note that if Δ_s is negative, s is unfeasible and does not belong to S_{iK}^1 .

The dynamic programming method exploits the fact that we need to compute solutions for all $K \subseteq \mathcal{N}_i$ and all $d \in \{1, \dots, D\}$. Thus, calculated optimal solutions can be combined to generate feasible solutions for problems involving additional machines.

For instance, let $X, Z \subseteq \mathcal{N}_i$ such that $X \cap Z = \emptyset$. Let $d \in \{2, \dots, D\}$. Let $x^* \in S_{iX}^1$ and $z^* \in S_{iZ}^{d-1}$ be already calculated optimal solutions. Then, assigning x^* on one machine and z^* on $d - 1$ machines, we obtain a feasible solution for the scheduling problem with d machines and the set of jobs $X \cup Z$. We note a solution obtained from combining two solutions x^* and z^* as (x^*, z^*) . Although solution (x^*, z^*) might not be an optimal solution for the problem, combining solutions in such a way ensures that we enumerate only promising feasible solutions.

More generally, we write $s = (u, v)$, the solution obtained from combining solution u and solution v . We have, $m_{(u,v)} = \max(m_u, m_v)$ and $\Delta_{(u,v)} = \min(\Delta_u, \Delta_v)$.

Let H_{iK}^d be the set of all such combinations. To avoid symmetries, we assume the single-machine solution to be the bottleneck machine inducing the makespan. Hence, we have:

$$H_{iK}^d = \left\{ (u, v) \mid u \in S_{iZ}^1, v \in S_{iK \setminus Z}^{d-1}, m_u \geq m_v \right\}$$

Minor improvements can reduce computation time. For example, the single-machine solution u will not appear in H_{iK}^d if it does not verify the equations $m_u \geq \frac{1}{d} \times \sum_{t \in J_K^i} p_t$ and $m_u \geq \max_{t \in J_K^i} p_t$, as the condition $m_u \geq m_v$ would never be verified.

A solution s dominates a solution s' if its makespan is lower and its maximum

starting time is higher. Formally, we define the solutions dominance as follows:

$$\begin{aligned} s \succeq s' &\Leftrightarrow (m_s \leq m_{s'} \wedge \Delta_s \geq \Delta_{s'}) \\ s = s' &\Leftrightarrow (m_s = m_{s'} \wedge \Delta_s = \Delta_{s'}) \\ s \succ s' &\Leftrightarrow (s \succeq s' \wedge s \neq s') \end{aligned}$$

We now describe the dynamic program's formula to calculate a set S_{iK}^d .

$$S_{iK}^d = \{s \in H_{iK}^d, \quad \forall s' \in S_{iK}^d \setminus \{s\} \quad s' \not\preceq s\} \quad (23)$$

We recall that S_{iK}^d contains the Pareto front minimizing m_s and maximizing Δ_s . When two solutions on the Pareto front have identical objective values, either can be chosen without loss of generality. We remove one arbitrarily. Algorithm 3 describes the procedure to calculate S_{iK}^d .

Algorithm 3 *compute* $S(d \in \{2, \dots, D\}, i \in V, K \subseteq \mathcal{N}_i)$

```

1:  $S \leftarrow \{\}$ 
2: for  $X \subset K$  do
3:   for  $x \in S_{iX}^1$  do
4:      $Z \leftarrow K \setminus X$ 
5:     for  $z \in S_{iZ}^{d-1}$  do
6:       if  $\nexists s \in S \mid s \succ (x, z)$  then
7:          $S \leftarrow S \cup \{(x, z)\}$ 
8: return  $S$ 

```

Note that with specific jobs' deadlines and processing times, the number of solutions in a set S_{iK}^d can also be exponential. In practice, the size of the sets remained manageable throughout all processed instances.

The algorithm is exponential in the size of K . However, as the output size can be exponentially larger than the input size, the algorithm is output-sensitive. The complexity of the algorithm computing algorithm 3 for each $i \in V, d \in \{2, \dots, D\}$ and $K \subseteq \mathcal{N}_i$ is polynomial in the size of its output.

5.2.3. Pareto front enumeration example

In this section, we present a simple example of enumeration using the dynamic program. Figure 2 represents a customer and its drone-reachable neighborhood. We consider a truck parked at customer i . Customers 1, 2, and 3 belong to the drone-reachable neighborhood of i ($\mathcal{N}_i = \{1, 2, 3\}$). Note that for readability reasons, different notations are used for the customers in the drone-reachable neighborhood and for the customer where the truck is parked. Figure 2 represents the one-way flight duration on each arc. Deadlines are provided above each customer node in \mathcal{N}_i . We consider $c_{0,i}^v$ to be zero, the service time for drones to be one, and the setup time to be zero for all customers.

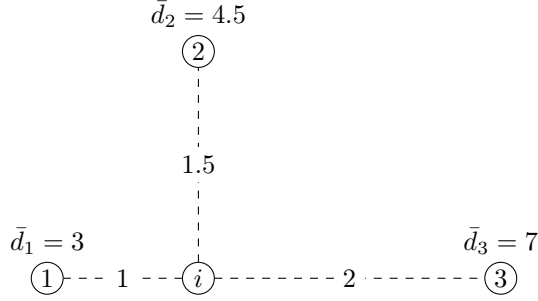


Figure 2: Example customer i and its drone-reachable neighborhood

In this example, we calculate $S_{i\{\mathbf{1},\mathbf{2},\mathbf{3}\}}^2$ using formula 23. In other words, we calculate the optimal drone schedules delivering a set of customers $\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$ using two drones from a truck parked at customer i for any truck arrival date.

We note a schedule with one machine as the order of the processed jobs between parentheses. For instance: $(\mathbf{1}, \mathbf{3})$ is the schedule with one machine, processing job $\mathbf{1}$, then job $\mathbf{3}$. We note a schedule with more than one machine as the order of each machine's processed jobs separated by a "|". For instance, $(\mathbf{3}|\mathbf{1}, \mathbf{2})$ is the schedule with two machines, where the first machine processes job $\mathbf{3}$ and the second machine processes job $\mathbf{1}$ and then job $\mathbf{2}$.

The first step of the algorithm is to model the problem as a scheduling problem. To convert the drone travel from customer i to customer v_t to a job t , we use equation 21 and equation 22. In table 3, the instance's data is summarized, and jobs are presented.

t	v_t	o_{v_t}	$c_{i,v_t}^d = c_{v_t,i}^d$	s_{v_t}	d_{v_t}	p_t	d_t
1	1	0	1	1	3	3	5
2	2	0	1.5	1	4.5	4	7
3	3	0	2	1	7	5	10

Table 3: Drone round-trips conversion to jobs

The second step is to sort the jobs by increasing deadlines in the ordered set $K \subseteq \mathcal{N}_i$. For clarity, this order is the lexicographic order here. The third step is to solve S_{iK}^1 with the Earliest Deadline First rule.

Formulas for the makespan and the maximal starting time are as follows:

$$m_{s_K} = \sum_{t \in J_K^i} p_t$$

$$\Delta_{s_K} = \min_{J_K^i} \left(\bar{d}_t - \sum_{\substack{t' \in J_K^i \\ C_{t'}^{s_K} \leq C_t^{s_K}}} p_{t'} \right)$$

$$\begin{aligned} S_{i\{1\}}^1 &= \{(\mathbf{1}) : m_{(\mathbf{1})} = 3, \Delta_{(\mathbf{1})} = 2\} \\ S_{i\{2\}}^1 &= \{(\mathbf{2}) : m_{(\mathbf{2})} = 4, \Delta_{(\mathbf{2})} = 3\} \\ S_{i\{3\}}^1 &= \{(\mathbf{3}) : m_{(\mathbf{3})} = 5, \Delta_{(\mathbf{3})} = 5\} \\ S_{i\{1,2\}}^1 &= \{(\mathbf{1}, \mathbf{2}) : m_{(\mathbf{1},2)} = 7, \Delta_{(\mathbf{1},2)} = 0\} \\ S_{i\{1,3\}}^1 &= \{(\mathbf{1}, \mathbf{3}) : m_{(\mathbf{1},3)} = 8, \Delta_{(\mathbf{1},3)} = 2\} \\ S_{i\{2,3\}}^1 &= \{(\mathbf{2}, \mathbf{3}) : m_{(\mathbf{2},3)} = 9, \Delta_{(\mathbf{2},3)} = 1\} \\ S_{i\{1,2,3\}}^1 &= \{(\mathbf{1}, \mathbf{2}, \mathbf{3}) : m_{(\mathbf{1},2,3)} = 12, \Delta_{(\mathbf{1},2,3)} = -2\} \end{aligned}$$

$S_{i\{1,2,3\}}^1$ is infeasible as $\Delta_{1,2,3} < 0$.

The fourth step is to compute $S_{i\{1,2,3\}}^2$ using the formula 23.

$$\{(u, v) | u \in S_{iZ}^1, v \in S_{i\{1,2,3\} \setminus Z}^1 \quad \forall Z \subseteq \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}\} = \{((\mathbf{1}), (\mathbf{2}, \mathbf{3})), \\ ((\mathbf{2}), (\mathbf{1}, \mathbf{3})), \\ ((\mathbf{3}), (\mathbf{1}, \mathbf{2}))\}$$

With this set, each combination of solutions is calculated using the values in step four. For instance $((\mathbf{1}), (\mathbf{2}, \mathbf{3}))$, $\mathbf{1}$ is processed on the first machine, while $\mathbf{2}$ and $\mathbf{3}$ are processed on the remaining machine, as $d - 1 = 1$. Their makespan and maximal starting time are calculated as follows:

$$\begin{aligned} m_{(\mathbf{1}|\mathbf{2},\mathbf{3})} &= \max(m_{\mathbf{1}}, m_{\mathbf{2},\mathbf{3}}) = 9 \\ \Delta_{(\mathbf{1}|\mathbf{2},\mathbf{3})} &= \min(\Delta_{\mathbf{1}}, \Delta_{\mathbf{2},\mathbf{3}}) = 1 \\ m_{(\mathbf{2}|\mathbf{1},\mathbf{3})} &= \max(m_{\mathbf{2}}, m_{\mathbf{1},\mathbf{3}}) = 8 \\ \Delta_{(\mathbf{2}|\mathbf{1},\mathbf{3})} &= \min(\Delta_{\mathbf{2}}, \Delta_{\mathbf{1},\mathbf{3}}) = 2 \\ m_{(\mathbf{3}|\mathbf{1},\mathbf{2})} &= \max(m_{\mathbf{3}}, m_{\mathbf{1},\mathbf{2}}) = 7 \\ \Delta_{(\mathbf{3}|\mathbf{1},\mathbf{2})} &= \min(\Delta_{\mathbf{3}}, \Delta_{\mathbf{1},\mathbf{2}}) = 0 \end{aligned}$$

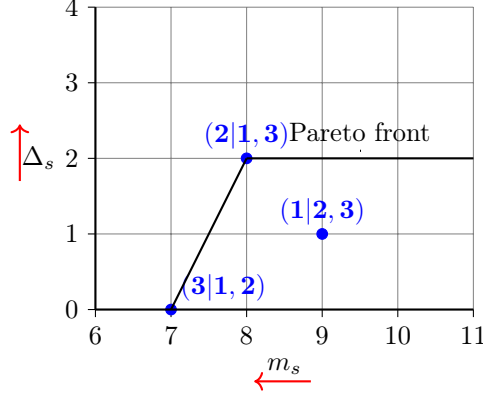


Figure 3: Pareto front for set $S_{i\{1,2,3\}}^2$

Figure 3 represents the Pareto front for $S_{i\{1,2,3\}}^2$ in the objective space. The red arrows represent the optimization direction for each objective. We observe that $(\mathbf{1}|\mathbf{2}, \mathbf{3})$ is dominated by $(\mathbf{2}|\mathbf{1}, \mathbf{3})$ as $m_{(\mathbf{1}|\mathbf{2}, \mathbf{3})} > m_{(\mathbf{2}|\mathbf{1}, \mathbf{3})}$ and $\Delta_{(\mathbf{1}|\mathbf{2}, \mathbf{3})} < \Delta_{(\mathbf{2}|\mathbf{1}, \mathbf{3})}$. Therefore, $(\mathbf{1}|\mathbf{2}, \mathbf{3})$ does not belong in $S_{i\{1,2,3\}}^2$. The two remaining solutions are not dominated and belong to $S_{i\{1,2,3\}}^2$. Finally, we have

$$S_{i\{1,2,3\}}^2 = \{(\mathbf{2}|\mathbf{1}, \mathbf{3}), (\mathbf{3}|\mathbf{1}, \mathbf{2})\}$$

In conclusion, to deliver customers 1, 2, and 3 from i using two drones:

- If the truck arrives at most at time 0, the optimal solution is to use a drone to deliver customer 3, and the other drone to deliver customer 1 and then customer 2. In that case, the makespan of the deliveries is 7.
- If the truck arrives at most at time 2, the optimal solution is to use a drone to deliver customer 2, and the other drone to deliver customer 1 and then customer 3. In that case, the makespan of the deliveries is 8.
- If the truck arrives after 2, there is no feasible solution.

A final remark is that if the truck travel time between the depot and customer i is strictly greater than 0 ($c_{(0,i)}^t > 0$), then the drone schedule $(\mathbf{1}, \mathbf{2})$, in which a single drone serves customer 1 followed by customer 2, would not be included in S_{iK}^1 , as the truck could not arrive before the schedule's maximal starting time ($\Delta_{(\mathbf{1}, \mathbf{2})} = 0$). In that case, $((\mathbf{3}), (\mathbf{1}, \mathbf{2}))$ would not be computed in the set of possible combinations for $S_{i\{1,2,3\}}^2$, resulting in $(\mathbf{3}|\mathbf{1}, \mathbf{2})$ not belonging in $S_{i\{1,2,3\}}^2$.

5.2.4. Criteria for valid arcs elimination

In this section, we introduce two criteria used in preprocessing to eliminate infeasible and dominated arcs in A^{dD} , from a subproblem graph \mathcal{G} , $d \in \{0, \dots, D\}$.

The first criterion is based on a maximum weight argument. Any schedule in S_{iK}^d , $d \in \{0, \dots, D\}$, $i \in V$, $K \subseteq \mathcal{N}_i$ does not correspond to a feasible drone schedule if the total payload is higher than the truck capacity, i.e. $q_i + d \times q_1^d + \sum_{k \in K} q_k > Q$ because the scheduling solution is not feasible in the routing problem, thus the arc representing this schedule is infeasible.

The second criterion is based on a replacement argument. Let a truck with d drones be parked at customer i . Suppose there is a faster route (combining truck and drones) serving a set of customers K than serving all of them by drone from i (i.e., using an appropriate drone schedule in S_{iK}^d). In that case, we know that the corresponding solution in S_{iK}^d will not be used in any optimal solution, and we can remove it safely. More formally, for a given $i \in V$, $d \in \{0, \dots, D\}$, $K \subseteq \mathcal{N}_i$ and $j \in V \setminus K \cup \{i\}$, let $\text{SPR}(d, i, K, j)$ be the duration of the shortest partial route from i to j using a truck equipped with d drones and serving either by drone or by truck the customers of set K . If for any feasible j , $\text{SPR}(d, i, K, j) < m_s + c_{i,j}^v$, s will not be used in an optimal solution.

We consider the following set of feasible j :

$$\left\{ j \in V_+ \setminus K \cup \{i\} \mid (c_{0,i}^v + \max(s_i^v, m_s) + c_{i,j}^v < \bar{d}_j) \wedge \left(q_i + d \times q_1^d + q_j + \sum_{k \in K} q_k \leq Q \right) \right\} \quad (24)$$

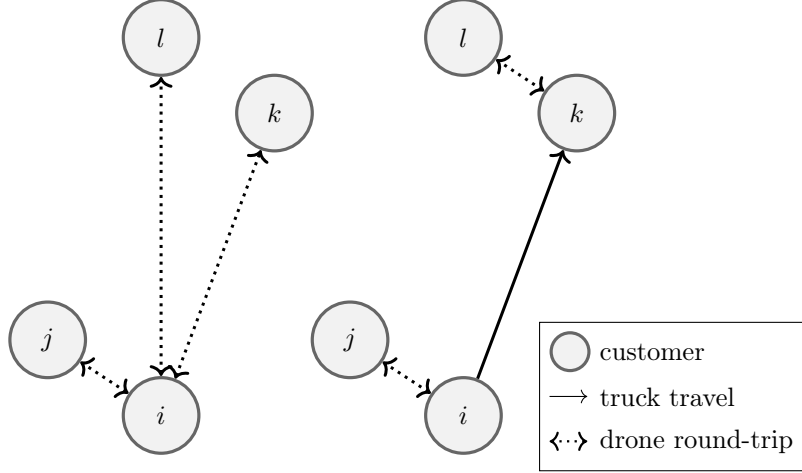


Figure 4: Context for an application of SPR criterion with one drone.

Figure 4 presents the intuition behind the SPR criterion. On the left side, a single drone serves customers j, k, l from customer i . On the right side, a drone serves customer j from customer i , then the truck moves to customer k , and then the drone serves customer l from customer k . It may be possible that the total duration of the left-side solution is longer than that of the right-side

solution. In that case, if any solution using the left-side deliveries can replace it with the right-side deliveries without diminishing the objective value while remaining feasible, the left-side deliveries do not have to be considered, and we can remove $S_{i\{jkl\}}^1$.

6. Heuristic branch-cut-and-price

In this section, we propose a heuristic branch-cut-and-price based on reducing the size of the subproblems graphs. It has four parameters :

D_h — The maximum number of drones a truck can carry.

w_h — The maximum number of customers that can be served by drone from a customer in a schedule.

k_h — The maximum number of drone-reachable neighbors. Only the k_h closest drone-reachable neighbors are kept.

p_h — The number of schedule with earliest deadline kept for each S_{iK}^d .

D_h decreases the number of subproblems. w_h and k_h reduce the number of drone arcs in each subproblem graph. Note that the difference between w_h and k_h is that k_h controls which customer can be served by drone from a parked vehicle, while w_h controls how many customers can be served by drone from a parked vehicle. Finally, p_h reduces the number of schedules and, therefore, the number of arcs in subproblems with multiple drones.

Let \mathcal{N}_i^k be the set of the k closest to i customers in \mathcal{N}_i . More formally, $\mathcal{N}_i^k \subseteq \mathcal{N}_i$, $|\mathcal{N}_i^k| = k$ and $\forall k \in \mathcal{N}_i^k, k' \in \mathcal{N}_i \setminus \mathcal{N}_i^k, c_{ik}^d \leq c_{ik'}^d$. Let S_{iK}^{dp} be the set of the p_h solutions from S_{iK}^d with earliest deadlines. To apply these rules, we transform each subproblems graphs $\mathcal{G}^d = (\mathcal{V}^d, \mathcal{A}^d), d \in \{1, \dots, D\}$ with $\mathcal{A}^d = A^T \cup A^D$ from the exact branch-cut-and-price subproblems graphs as follows: we create the graphs $\mathcal{G}_h^d = (\mathcal{V}_h^d, \mathcal{A}_h^d), d \in \{1, \dots, D_h\}$, where $\mathcal{V}_h^d = \mathcal{V}^d$ and $\mathcal{A}_h^d = A^T \cup \{(i, i')^s \in A^D, \forall K \subset \mathcal{N}_i^{k_h}, s \in S_{iK}^{dp_h} : |K| \leq w_h\}$

7. Computational experiments

7.1. Configuration

Our algorithms, such as the dynamic program for the enumeration, the separation algorithm for the ARCC, and the SPR criterion, are coded in C++. We also used:

- IBM ILOG CPLEX 20.1 as the LP solver in the relaxed restricted master problem and as the solver for the enumerated MIPs.

- The generic branch-cut-and-price code from (Sadykov and Vanderbeck, 2021), with automatic dual price smoothing stabilization from (Pessoa et al., 2018), primal heuristics from (Sadykov et al., 2019). We also use the labeling algorithm code from (Sadykov et al., 2021).

The experiments presented in this paper were carried out:

- Using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d’Aquitaine (see <https://www.plafrim.fr>).
- Running on 2.6 GHz 2x 18-core Cascade Lake Intel Xeon Skylake Gold 6240 CPU using a single thread with 5.3 Go memory.

7.2. Instances

To measure the performance of our algorithm, we used and extended the sets of instances from (Zhou et al., 2023), which is composed of modified instances from (Chen et al., 2021), and from (Solomon, 1987). In all these instances, the trucks and the drones follow the same distance metric. We also used clustered instances from (Gehring and Homberger, 1999) to experiment on larger instances, as they are an extension of the Solomon (1987) instances. The maximal drone capacity is set to 10 kg as in (Chen et al., 2021), the drone service times are set to $s_i^d = \lfloor 0.5s_i^v \rfloor, i \in V$.

In the instances proposed in (Chen et al., 2021), an additional $\lfloor 0.2 \times |V| \rfloor$ customers are randomly selected to be unavailable for drone delivery. Table 4 describes the different data sets. For each data set, we give the name of the instances (Instances) where X is the id of a particular instance of the set, the number of instances in the set (Size), the customer positioning (Pos), the distance metric (Dist), the original authors of the instances (Authors), the capacity of the vehicle (Q), the maximum number of drones allowed on a vehicle (D), the total number of vehicles (k_v) and the total number of drones (k_d). In column (Pos), "Rand" stands for "Random", and "Clust" for "Clustered". In column (Dist), "Eucl" stands for "Euclidean."

Instances	$ V $	Size	Pos	Dist	Authors	Q	D	k_v	k_d
Small-Size Instances									
<i>Cardiff_10_X</i>	10	20	Rand	Real	Chen et al. (2021)*	170	2	4	4
<i>Cardiff_15_X</i>	15	20	Rand	Real	Chen et al. (2021)	170	2	5	5
<i>Cardiff_25_X</i>	25	20	Rand	Real	Chen et al. (2021)	200	3	5	10
<i>c10X_35</i>	35	9	Clust	Eucl	Solomon (1987)	250	3	5	10
Medium-Size Instances									
<i>Cardiff_40_X</i>	40	20	Rand	Real	Chen et al. (2021)*	200	3	5	10
<i>c10X_100</i>	100	9	Clust	Eucl	Solomon (1987)	250	3	14	28
<i>120_C1_2_X</i>	120	10	Clust	Eucl	Gehring and Homberger (1999)	250	3	40	60
<i>130_C1_2_X</i>	130	10	Clust	Eucl	Gehring and Homberger (1999)	250	3	40	60
<i>140_C1_2_X</i>	140	10	Clust	Eucl	Gehring and Homberger (1999)	250	3	40	60
<i>150_C1_2_X</i>	150	10	Clust	Eucl	Gehring and Homberger (1999)	250	3	40	60

Table 4: Instances sets and their characteristics

Note that the set with 10 customers and the set with 40 customers are marked with ‘*’ as they are not in (Chen et al., 2021), but are derived from sets with a larger number of customers from (Chen et al., 2021), namely 15 and 50, by keeping only the 10 and 40 first customers.

The instances *c10X_N* are the 100-customers clustered instances called *c10X* in (Solomon, 1987), restricted to the N first customers. We categorize instances with 35 or fewer customers as small, while instances with more than 35 customers will be referred to as medium. The instances $N_C1_2_X$ are the 200-customer clustered instances called *C1_2_X* in (Gehring and Homberger, 1999), restricted to the first N customers.

7.3. Results

This section presents all the results obtained by setting the computation time limit to three hours per instance. Unless specified otherwise, the ARCC and the SPR criterion are used. To measure the difficulty of an instance more precisely, we introduce $\hat{K} = \max_{i \in V} |\mathcal{N}_i|$ the largest drone-reachable neighborhood of an instance, and $|\mathcal{A}^{\max}| = \max_{d \in D} |\mathcal{A}^d|$ the maximum number of arcs in a subproblem graph. $|\mathcal{A}^{\max}|$ is strongly dependent on \hat{K} , as the number of arcs in a subproblem graph is exponential in the size of the drone-reachable neighborhoods. The gap is calculated as follows: $\frac{PB-DB}{DB}$, with PB the primal bound and DP the dual bound.

In the first subsection, we present a comparison with the state-of-the-art results on the small-size instances. In the second subsection, we present the performance of our algorithm on medium-sized instances. In the third subsection, we present the results of the heuristic branch-cut-and-price on instances with large drone-reachable neighborhoods.

7.3.1. Performance of the exact BCP on the small-size instances

Table 5 shows the summary of the performance of our algorithm on the small-size instances and a comparison with the results obtained in (Zhou et al., 2023). The comparison is fair, as we use a computer with similar speed. For each data set, we give the name of the set (Instances), the number of instances solved optimally over the total number instances in the dataset (Solved), the average time in seconds required to solve the instances of the dataset (Time), the average number of nodes processed in the branch-cut-and-price algorithm (Nodes), and the maximum number of arcs in the subproblems graphs ($|\mathcal{A}^{\max}|$).

Instances	Zhou et al.		Our			
	Solved	Time (s)	Solved	Time (s)	Nodes	$ \mathcal{A}^{\max} $
<i>Cardiff_10_X</i>	20/20	0.4	20/20	0.57	1	217
<i>Cardiff_15_X</i>	20/20	0.7	20/20	1.33	1	993
<i>Cardiff_25_X</i>	15/20	3561.5	20/20	86.25	2.5	5207
<i>c10X_35</i>	4/9	7408	9/9	556.30	8.12	2490

Table 5: Average exact performance comparison given small-size instances

Our branch-price-and-cut algorithm can solve the small-size instances very efficiently. Indeed, it solves all instances featured in (Zhou et al., 2023), including 10 solved for the first time. The computation time is improved by a factor of 13 for the largest instances of the set. The number of nodes processed is very low. Our algorithm solves any small instance in less than half an hour. Therefore, we increase the difficulty of the instances by adding customers.

7.3.2. Performance of the exact BCP on the medium-size instances

Table 6 shows the summary of the performance of our algorithm on the medium-size instances. For each data set, we give the name of the set (Instances), the maximum number of arcs in the subproblems graphs ($|\mathcal{A}^{\max}|$) of all instances, the average maximal size of the drone-reachable neighborhoods ($\hat{\mathcal{K}}$), the number of instances solved optimally over the total number instances (Solved), the number of instances for which no feasible solution has been found (No Sol), the average Gap of the instances for which a feasible solution has been found, the average time in seconds required to solve the instances of the dataset (Time) and the average number of nodes processed in the branch-cut-and-price algorithm (Nodes). The detailed results can be found in Table 11, Table12, Table 13, and Table 14 in Appendix B.

Instances	$ \mathcal{A}^{\max} $	$\hat{\mathcal{K}}$	Solved	No Sol	Gap	Time (s)	Nodes
<i>Cardiff_40_X</i>	78422.55	13	9/20	6	3.12	6241.96	12.2
<i>c10X_100</i>	19009.23	11	8/9	0	0.14	3134.03	30.4
<i>120_C1_2_X</i>	14931.50	3	7/10	3	0	4430.94	58.8
<i>130_C1_2_X</i>	17615.80	4	5/10	4	0.07	7081.01	77.4
<i>140_C1_2_X</i>	20367.80	4	3/10	4	0.27	8229.01	84.3
<i>150_C1_2_X</i>	23332.80	4	2/10	4	0.45	9384.44	107.8

Table 6: Average exact performance comparison given medium-size instances

The size of the drone-reachable neighborhood significantly impacts the number of arcs of the subproblems graphs, which strongly affects calculation time. Therefore, the size of the drone-reachable neighborhood should be considered when measuring the size of an instance. Our algorithm is particularly efficient for solving clustered instances: it solves most clustered instances with 120 customers and some with up to 150 customers. It is worth mentioning that in Gehring and Homberger (1999) instances, due to the weight of the packages, less than 50% of customers are available for drone delivery, resulting in limited size drone-reachable neighborhoods. Increasing the number of customers in Gehring and Homberger (1999) instances slightly increases the gap and the number of processed nodes and decreases the number of instances solved to optimality. However, with randomized instances, the performance of our method deteriorates much faster. With 40 customers, the algorithm can only find optimal solutions for half of the Chen et al. (2021) instances and cannot find any feasible solution for 6 instances. The modest performance can be attributed to the large and highly interconnected drone-reachable neighborhoods in the randomized instances, leading to an important number of set S_{iK}^d . However, even in these instances, the number of solutions in each S_{iK}^d remained tractable. Note that instances *Cardiff_40_X*, in addition to having a high number of arcs on average, also have significant disparity among the instances of the set, as the standard deviation of the maximum number of arcs in a subproblem graph is 61705.8. We only solve less than half instances of *Cardiff_40_X* dataset. Due to the extremely high number of arcs, solving a node is slow. We propose to use the heuristic defined in section 6 to obtain good quality solutions.

7.3.3. Performance of the heuristic BCP on the random medium-size instances

To evaluate the performance of the proposed heuristic, we tested several different settings, which are reported in Table 7. The last row of the table corresponds to the performance of the exact BCP algorithm. In the absence of results in the literature, we compare all the results obtained by both the exact BCP and the heuristic BCP and retain the BKS from one of these methods. The average gap between the heuristic solution and the BKS is then calculated as follows: $\frac{\text{PB}-\text{BKS}}{\text{BKS}}$, with PB the primal bound and BKS the best-known solution value (Gap BKS).

Table 7 shows the summary of the performance of our heuristic branch-cut-

and-price on the Chen et al. (2021) instances with different sets of parameters. For each set of parameters D_h , w_h , k_h , and p_h , we give the average number of arcs in the largest subproblem of each instance ($|\mathcal{A}^{\max}|$), the average number of nodes processed in the branch-cut-and-price algorithm (Nodes), the average gap between the best-known solution and the solution found in percentage for the instances where a solution was found (Gap BKS), the number of instances for which the solver reached the time limit (Time lim), the number of instances for which the solver was not able to find a feasible solution (No sol) and the average solving time in seconds (Time). When the value of p_h is inf, it means the parameter is left unrestricted. The detailed results can be found in table 15 in Appendix B.

Note that the maximum authorized number of drones per truck in these instances is 3, and the largest drone-reachable neighborhood is 15; hence, setting the related parameters to these values is similar to leaving them unrestricted.

D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Time lim	No sol	Gap BKS	Time (s)	Nodes
2	2	2	1	1796.30	0	0	4	392.72	5
2	3	3	1	1937.20	0	0	2.09	510.60	7.1
2	5	5	1	2641.55	0	0	1.05	360.44	5.6
2	8	5	1	6250.15	6	0	0.74	3738.86	19.4
3	5	3	1	2490.50	0	0	0.85	681.09	9.7
3	5	5	1	2664.95	0	0	0.67	381.38	5.4
3	5	5	inf	2773.70	0	0	0.67	348.23	5.1
3	8	5	1	6465.15	4	1	0.31	3496.81	25.2
3	8	5	inf	7767.95	6	2	0.26	3878.18	20.6
3	8	8	1	7088.75	6	2	0.18	4052.90	23.2
3	15	10	1	15222.95	5	0	0.04	4125.97	29.3
3	15	10	inf	78192.80	11	8	2.49	6203.80	12.5
3	15	15	1	43094.35	9	7	0.44	5625.45	13.5
3	15	15	inf	78422.55	11	6	1.89	6241.96	13

Table 7: Heuristic average performances given parameters on the Chen et al. (2021) instances with 40 customers

We can observe in table 7 that reducing the number of arcs in \mathcal{A}^{dD} with $d \in \{1, \dots, D_h\}$ by decreasing w_h or k_h can significantly decrease computation time, without substantially decreasing the quality of the solutions. Reducing D_h has a significant negative impact on the gap BKS without decreasing the computation time for these instances. It is worth mentioning that the heuristic with parameters $D_h = 3$, $w_h = 15$, $k_h = 10$, and $p_h = 1$ gives a major improvement compared to the exact method, as both the average gap and the average computation time largely decrease. The heuristic with parameters $D_h = 3$, $w_h = 5$, $k_h = 5$, $p_h = 1$, despite having a slightly larger gap, exhibits a swift average solving time of less than 10 minutes. Unrestricted parameter p_h significantly increases the number of arcs and the computation time. while having no positive

impact on the gap. That is why we set it to one in most cases.

7.4. Sensitivity analysis

In this section, we first present the impact of the ARCC and then present the impact of the SPR criterion. We use the concept of cut-off values. Cut-off values are primal bounds that the user provides to the algorithm to prune the branch and bound tree. We use the best-known solution as a cut-off value. Although these values can be used to prove solution optimality, the goal here is to prevent random performance of the primal heuristics, enabling a fair comparison of performance with and without the tested improvement. Unless specified otherwise, the ARCC and the SPR criterion are used. We recall that the gap is calculated as follows: $\frac{PB-DB}{DB}$, with PB the primal bound and DP the dual bound.

7.4.1. Impact of the adapted rounded capacity cuts

This section presents the impact of the adapted rounded capacity cuts. The capacity constraint in the Chen et al. (2021) instances is not tight, as a substantial proportion of the truck’s capacity remains unused in an optimal solution. Therefore, the ARCC are not helpful for Chen et al. (2021) instances. In addition, the instances with 100 customers are all solved within 10 minutes, most of them solved at the root node. For these reasons, we focus on the instances from Gehring and Homberger (1999) in this section. As shown in table 6, within the three-hour limit, the exact branch-cut-and-price does not find a feasible solution for several instances of the Gehring and Homberger (1999) datasets. We have run the exact BCP on these instances with a ten-hour time limit to find accurate cut-off values. However, we did not find a feasible solution for all instances with 150 customers. Therefore, we present the results on the Gehring and Homberger (1999) instances with 120, 130, and 140 customers in table 8. We compare the performances with and without the ARCC. Table 8 shows the experiments’ summary to measure the ARCC’s impact. For each data set, we give the name of the set (Instances), whether the ARCC were used or not (ARCC), the average gap obtained on instances that are not solved optimally within the time limit (Gap), the average time in seconds required to solve the instances of the dataset (Time) and the average number of nodes processed in the branch-cut-and-price algorithm (Nodes). Note that the number of optimal solutions found is omitted as the ARCC does not impact this number. The detailed results can be found in the table 16 in Appendix B.

Instances	ARCC	Gap	Time (s)	Nodes
<i>120_C1_2_X</i>	yes	0.12	2437.30	31.6
<i>120_C1_2_X</i>	no	0.12	2451.13	40.6
<i>130_C1_2_X</i>	yes	0.28	5784.12	88.2
<i>130_C1_2_X</i>	no	0.30	5744.47	67.6
<i>140_C1_2_X</i>	yes	0.56	5955.80	75.4
<i>140_C1_2_X</i>	no	0.61	5925.10	72.6

Table 8: Average impact of the adapted rounded capacity cuts on the (Gehring and Homberger, 1999) instances

We can observe in table 8 that the ARCC have a positive impact on the gap, growing with the size of the instances. The ARCC have a negligible impact on the computation time because the capacity constraints are not very tight for these instances.

7.4.2. Impact of the shorter partial route criterion

In this section, we study the impact of the shorter partial route criterion in terms of the exact algorithm’s efficiency and graph size. In our computational experiment, and because checking this criterion would be too time-consuming, we only consider special cases where the truck can only visit a single customer of K , and the others must be delivered by drone. As the instances from (Gehring and Homberger, 1999) have a small drone-reachable neighborhood, this criterion does not significantly impact the number of arcs. For this reason, we perform the sensibility analysis only for the medium-size instances from (Chen et al., 2021) and (Solomon, 1987). We compare the performances of the algorithm with and without the SPR criterion. We recall that \mathcal{G}^d is the directed graph representing the delivery possibilities for a truck with d drones. Table 9 shows the experiments’ summary to measure the SPR criterion’s impact on the medium-size instances. For each data set, we give the name of the set (Instances), the number of arcs in subproblem graph and the proportion of arcs compared to without the SPR criterion ($|\mathcal{A}^1|$), ($|\mathcal{A}^2|$), ($|\mathcal{A}^3|$), the average Gap of the instances for which a feasible solution has been found, the average time in seconds required to solve the instances of the dataset (Time) and the average number of nodes processed in the branch-cut-and-price algorithm (Nodes).

Instances	SPR	$ \mathcal{A}^1 $	$ \mathcal{A}^2 $	$ \mathcal{A}^3 $	Gap	Time (s)	Nodes
<i>Cardiff_40_X</i>	yes	4449.15 (20%)	62169.85 (54%)	72709.35 (97%)	6.84	6376.47	10.7
<i>Cardiff_40_X</i>	no	22597.45 (100%)	116217.80 (100%)	75056.20 (100%)	7.26	6318.14	8.8
<i>c10X_100</i>	yes	12902 (68%)	18298.67 (95%)	19009.23 (100%)	0	206.81	5.23
<i>c10X_100</i>	no	19079 (100%)	19326.56 (100%)	19009.23 (100%)	0	274.63	5.89

Table 9: Sensitivity analysis over SPR criterion

Note that with cut-off values, the algorithm solves all *c10X_100* instances to optimality. The criterion removes a substantial number of arcs in \mathcal{G}^1 in both

cases, less in \mathcal{G}^2 , and almost none in \mathcal{G}^3 . It removes more arcs when instances have larger drone-reachable neighborhoods.

However, while the computation efficiency improvement is mild, the SPR criterion slightly accelerates computation time in most solved instances for both data sets. The unsolved instances gaps are decreased by 0.5% on average. Although the criterion seems to have a negative impact on computation time in the *Cardiff_40_X* instances, the time decreases for 7 out of 9 instances solved optimally. Also, the criterion decreases the number of nodes in the *c10X_100* instances. The impact is positive, as it improves convergence speed. The criterion increases the number of nodes in instances *Cardiff_40_X*, which is also a positive result, as it is mainly due to the number of nodes in instances where the optimal solution was not found, demonstrating a faster node processing.

7.4.3. Impact of the restricted master heuristic

In this section, we present the impact of the restricted master heuristic. In this experiment, cut-off values are not provided to the solver.

Instances	RMH	Solved	No Sol	Gap	Time	Nodes
<i>Cardiff_40_X</i>	yes	9/20	6	3.12	6241.96	12.2
<i>Cardiff_40_X</i>	no	11/20	2	3.64	5621.62	39.0
<i>c10X_100</i>	yes	8/9	0	0.14	3134.03	30.4
<i>c10X_100</i>	no	8/9	1	0.00	4184.41	57.0
<i>120_C1_2_X</i>	yes	7/10	3	0	4430.94	58.8
<i>120_C1_2_X</i>	no	6/10	2	0.19	6116.54	132.40

Table 10: Sensitivity analysis over restricted master heuristic.

It can be observed that the absence of the restricted master heuristic leads to faster node processing. Removing the restricted master heuristic positively impacts the results on the instances *Cardiff_40_X*. This can be explained by the fact that elementary columns are challenging to obtain in these instances due to the large drone-reachable neighborhoods and the lack of well-defined clusters. Consequently, the restricted master heuristic’s solution space may be very limited or even empty, potentially leading to negative effects.

7.5. Large instance solution illustration

In this section, we display an optimal solution for the *c100_02.txt* instance.

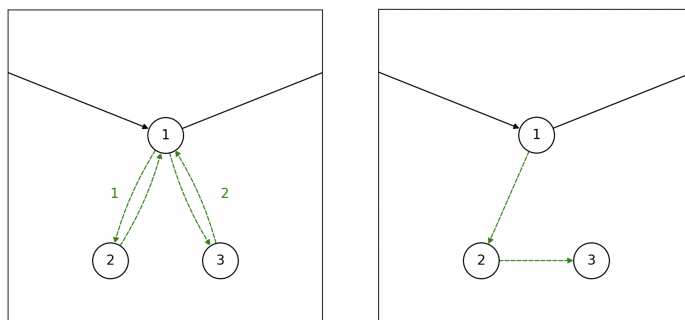


Figure 5: Drone routes (left) and Drone schedule (right) representations

Black arrows represent truck movement. Green dashed arrows represent a drone delivery. Each path of green arrows represents the delivery order of a single drone from the first customer of the path. Multiple green arrows from a customer illustrate multiple parallel drone deliveries.

Figure 5 visually explains the drone schedule representation. The diagram on the left depicts the sequence of movements along the drone route: a truck arrives at customer 1, a drone makes a first round-trip to deliver customer 2 (labeled 1 in green), then the same drone makes a second round-trip to deliver customer 3 (labeled 2 in green), and finally, the truck leaves customer 1. The diagram on the right depicts the drone schedule representation of the same sequence. The path of green arrows represents the sequence of round trips made from its source by a drone. Therefore, a single drone performs a round trip to customer 2 and then to customer 3.

We use this scheduling representation because it allows the representation of schedules without further annotations and significantly simplifies the visualization.

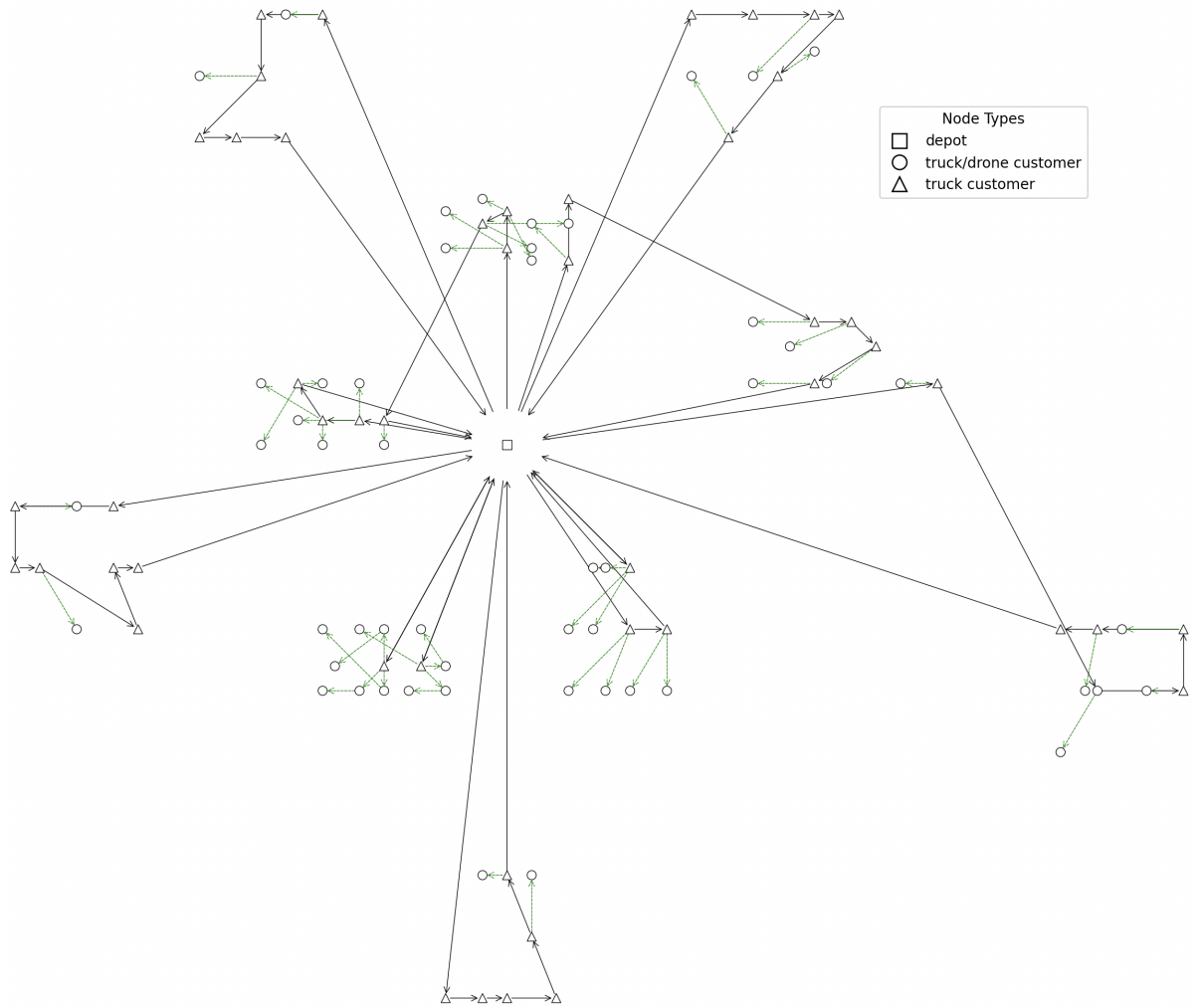


Figure 6: Optimal solution for c102_100.txt

8. Conclusion

In this paper, we introduced a new branch-cut-and-price algorithm for the two-echelon vehicle routing problem with drones. This algorithm is based on the enumeration of the drone's round trips between customers and the vehicle parked at another customer. We presented ways to improve the method by reducing the number of drone trips enumerated and by adapting the well-known rounded capacity cuts supported by sensitivity analysis.

Extensive numerical experiments demonstrated the high performance of our method. We solve all literature instances containing up to 35 customers, improving computation times significantly by a factor higher than 10, and optimally

solving 10 for the first time.

We also proposed 6 new sets of instances to promote research on VRP-D-type problems. The experiments on our method show that we solve all but one clustered instance with 100 customers and more than half of the clustered instances with 120 customers. In the random dataset with 40 customers and large drone-reachable neighborhoods, 11 instances remain open. To tackle these instances, we proposed a heuristic branch-cut-and-price approach based on eliminating certain arcs in the subproblems. This approach shows excellent results, as the gaps with the best-known solutions are minimal. We also show the improvement induced by adapting the rounded capacity cuts and the SPR criterion.

Testing these new instances highlighted some limitations of our branch-cut-and-price approach. In the largest instances, the size of the sub-problem graph becomes intractable due to the exponential number of parallel arcs representing drone round-trip possibilities. A possible remedy is to model the scheduling problem as part of the RCSP to avoid explicitly enumerating the scheduling solutions. This modification would require special care for the resources, as only one drone is responsible for the duration of the makespan. In the same vein, it would be helpful to find other rules of dominance for the drone arcs.

To enhance the speed of the exact algorithm, an alternative approach would be to develop a problem-specific primal heuristic aimed at quickly identifying high-quality primal bounds. It would also be beneficial to develop instances better suited for truck and drone deliveries, incorporating distinct metrics for drones and trucks, as well as realistic features related to last-mile delivery services.

One could also consider uncertainty regarding traffic congestion. It could also be interesting to compare the gain using the 2E-VRP-D and the VRPD on different sets of instances and objective functions. In addition, most national and metropolitan public authorities are moving towards tighter regulations for deliveries in city centers: more restrictions are being imposed on vehicle access times and, therefore, also on customer deliveries. One could also integrate time window constraints with non-zero ready times into our model. This extension will require special care for drone operations, as drones might need to wait between round trips.

Another perspective is to apply this type of enumeration strategy to problems with a similar structure, such as the park-and-loop vehicle routing problem. In this problem, the vehicle driver parked at a client can make a walking tour to serve customers, avoiding network congestion and gas use.

Appendix A Calculation of drone-reachable neighborhood

Let Q^d be the drone’s maximal capacity in *kg*. For $i \in V$, f_i^d is set to 0 if $q_i > Q^d$ or if i is not available for deliveries by drone. Otherwise, we set $f_i^d = 1$. We introduce a parameter \bar{f}_i^d expressing whether a truck parked at customer $i \in V$ can launch drones. A customer’s location might not be available for a drone launch if no parking spot is available nearby, for example. In this case, we set $\bar{f}_i^d = 0$ and $f_i^d = 1$ otherwise. An energy function limits the maximal flight

distance, calculated as follows: $P(q) = (W + m + q)^{\frac{3}{2}} \sqrt{\frac{g^3}{2\rho\zeta h}}$ in Watt, introduced by Dorling et al. (2016) as the hovering power consumption of a h-rotor drone, an upper bound on the general power consumption of a h-rotor drone, and based on Leishman (2006). W is the frame weight (kg), m the battery weight (kg), q the payload (kg), g is the force due to gravity (N), ρ the fluid density of air in (kg/m^3) and ζ the area of spinning blade (m^2). We note B_c the drone’s battery capacity in watt-hours. Let $g = 9.81$ N/kg and $\rho = 1.204$ kg/m³ as in (Dorling et al., 2016).

$$N_i = \{j \in V \mid (\bar{f}_i^d == 1) \wedge (f_j^d == 1) \wedge (P(q_i) * c_{i,j}^d + P(0) * c_{j,i}^d \leq B_c) \wedge (c_{0,i}^v + c_{i,j}^d \leq \bar{d}_j) \quad i \in V\}$$

Appendix B Detailed computational experiments

In the redfollowing tables we present the detailed results of our experiments. We present different information among: the name of the set (Instances), the maximum number of arcs in the subproblem \mathcal{G}^D ($|\mathcal{A}^D|$) of all instances, the average maximal size of the drone-reachable neighborhoods ($\bar{\mathcal{K}}$), the number of instances solved optimally over the total number instances (Solved), the number of instances for which no feasible solution have been found (No Sol), the average Gap of the instances for which a feasible solution have been found, the average time in seconds required to solve the instances of the dataset (Time) and the average number of nodes processed in the branch-cut-and-price algorithm (Nodes).

Table 11: Detailed performance of the exact algorithm on the small-size instances

Instance	$ \mathcal{A}^{\max} $	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
Cardiff_10_01	217	0	0.72	1	1223.62	1223.62	0	0.79
Cardiff_10_02	134	0	0.447	1	1905.08	1905.08	0	0.49
Cardiff_10_03	147	0	0.543	1	1412.67	1412.67	0	0.6
Cardiff_10_04	136	0	0.572	1	1633.32	1633.32	0	0.62
Cardiff_10_05	136	0	0.436	1	1682.93	1682.93	0	0.48
Cardiff_10_06	155	0	0.591	1	1221.75	1221.75	0	0.65
Cardiff_10_07	129	0	0.473	1	1931.75	1931.75	0	0.52
Cardiff_10_08	143	0	0.615	1	1578.15	1578.15	0	0.67
Cardiff_10_09	126	0	0.456	1	2552.03	2552.03	0	0.49
Cardiff_10_10	125	0	0.46	1	2045.18	2045.18	0	0.5
Cardiff_10_11	137	0	0.424	1	2095.25	2095.25	0	0.46
Cardiff_10_12	161	0	0.578	1	1737.25	1737.25	0	0.63
Cardiff_10_13	136	0	0.5	1	1485.88	1485.88	0	0.57
Cardiff_10_14	125	0	0.441	1	1926.62	1926.62	0	0.5
Cardiff_10_15	202	0	0.479	1	1814.33	1814.33	0	0.53
Cardiff_10_16	148	0	0.602	1	1140.05	1140.05	0	0.69

Continued on next page

Table 11: Detailed performance of the exact algorithm on the small-size instances

Instance	$ \mathcal{A}^{\max} $	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
Cardiff_10_17	129	0	0.449	1	1733.83	1733.83	0	0.5
Cardiff_10_18	137	0	0.456	1	1960.93	1960.93	0	0.51
Cardiff_10_19	136	0	0.478	1	1533.67	1533.67	0	0.53
Cardiff_10_20	135	0	0.502	1	1366.42	1366.42	0	0.59
Cardiff_15_01	993	0	3.59	1	1332	1332	0	3.66
Cardiff_15_02	383	0	0.773	1	1970.62	1970.62	0	0.83
Cardiff_15_03	292	0	1.099	1	1971.38	1971.38	0	1.19
Cardiff_15_04	320	0	0.799	1	2188.32	2188.32	0	0.84
Cardiff_15_05	311	0	0.727	1	1831.27	1831.27	0	0.79
Cardiff_15_06	408	0	1.395	1	1695.17	1695.17	0	1.5
Cardiff_15_07	318	0	0.755	1	2050.67	2050.67	0	0.82
Cardiff_15_08	318	0	1.091	1	1689.55	1689.55	0	1.17
Cardiff_15_09	322	0	0.687	1	2765.68	2765.68	0	0.74
Cardiff_15_10	271	0	0.9	1	2984.98	2984.98	0	0.94
Cardiff_15_11	319	0	0.824	1	2242.98	2242.98	0	0.89
Cardiff_15_12	379	0	0.991	1	2155.83	2155.83	0	1.06
Cardiff_15_13	369	0	1.512	1	1606.55	1606.55	0	1.59
Cardiff_15_14	300	0	4.598	1	2369.88	2369.88	0	4.76
Cardiff_15_15	728	0	1.228	1	1883.45	1883.45	0	1.29
Cardiff_15_16	522	0	1.129	1	1756.85	1756.85	0	1.22
Cardiff_15_17	293	0	0.583	1	1858.17	1858.17	0	0.64
Cardiff_15_18	330	0	0.893	1	2288.73	2288.73	0	0.99
Cardiff_15_19	303	0	0.69	1	1762.17	1762.17	0	0.76
Cardiff_15_20	306	0	0.871	1	1697.1	1697.1	0	0.95
Cardiff_25_01	1622	0	78.333	1	2233.87	2233.87	0	78.81
Cardiff_25_02	1688	0.3	47.381	3	1979.95	1979.95	0	64.68
Cardiff_25_03	985	0.7	28.089	3	2389.7	2389.7	0	40.9
Cardiff_25_04	1280	5.8	31.931	3	2797.63	2797.63	0	66.87
Cardiff_25_05	1481	0	27.47	1	2062.97	2062.97	0	27.72
Cardiff_25_06	1140	0	24.194	1	2075.48	2075.48	0	24.52
Cardiff_25_07	1212	1.3	31.748	3	2112.6	2112.6	0	44.93
Cardiff_25_08	1119	0	20.74	1	2605.48	2605.48	0	20.91
Cardiff_25_09	1634	1.4	33.46	3	2055.22	2055.22	0	48.93
Cardiff_25_10	1605	2.7	75.233	5	2217.12	2217.12	0	296.04
Cardiff_25_11	5207	0.7	85.909	3	2051.12	2051.12	0	99.0
Cardiff_25_12	1245	7	18.103	5	4775.93	4775.93	0	157.68
Cardiff_25_13	2730	0	57.655	1	2248.63	2248.63	0	58.04
Cardiff_25_14	900	4	57.477	3	2530.68	2530.68	0	257.38
Cardiff_25_15	1240	0.7	35.323	3	2871.6	2871.6	0	51.14
Cardiff_25_16	1553	1.5	29.461	3	2503.27	2503.27	0	40.64
Cardiff_25_17	3780	0	65.629	1	2275.83	2275.83	0	65.9

Continued on next page

Table 11: Detailed performance of the exact algorithm on the small-size instances

Instance	$ \mathcal{A}^{\max} $	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
Cardiff_25_18	922	0.6	56.438	3	2167.9	2167.9	0	125.93
Cardiff_25_19	1796	1.7	55.092	3	2257.8	2257.8	0	71.74
Cardiff_25_20	2449	0	82.744	1	2816.63	2816.63	0	83.14
c101_35	2460	0.7	20.556	3	1896.5	1896.5	0	113.09
c102_35	2460	0.5	27.488	5	1892.1	1892.1	0	729.18
c103_35	2460	0.6	17.262	3	1889.2	1889.2	0	32.16
c104_35	2472	0.8	50.63	25	1888.6	1888.6	0	1406.2
c105_35	2464	0.5	38.436	3	1896.5	1896.5	0	257.74
c106_35	2460	0.6	34.21	5	1896.5	1896.5	0	241.3
c107_35	2490	0.6	37.053	3	1896.5	1896.5	0	162.77
c108_35	2460	0.7	28.446	5	1893	1893	0	408.04
c109_35	2490	0.5	121.423	21	1893	1893	0	1656.22

Note that the instance *Cardiff_40_11* is not feasible under our assumptions.

Table 12: Detailed performance of the exact algorithm on instances adapted from (Chen et al., 2021)

Instance	$ \mathcal{A}^{\max} $	$\hat{\mathcal{K}}$	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
Cardiff_40_01	13034	10	0.3	214.127	3	2931.88	2931.88	0	306.84
Cardiff_40_02	95883	14	20.1	92.803	37	3044.72	3217.42	5.7	10800
Cardiff_40_03	30318	12	2.7	168.228	3	2682.5	2682.5	0	351.02
Cardiff_40_04	169680	15	-	401.227	9	4292.31	-	-	10800
Cardiff_40_05	25801	11	1.7	329.68	11	3770.9	3770.9	0	2141.71
Cardiff_40_06	26451	12	12.5	203.151	19	2909	2909	0	1633.04
Cardiff_40_07	89523	15	-	674.039	3	2093.42	-	-	10800
Cardiff_40_08	9510	9	0.6	191.295	3	3834.02	3834.02	0	358.6
Cardiff_40_09	81734	14	35.1	381.086	11	3580.1	4484.18	25.3	10800
Cardiff_40_10	55913	14	1.3	171.288	3	2692.08	2692.08	0	573.32
Cardiff_40_11	48327	12	0	29.336	1	unfeasible	unfeasible	0	30.12
Cardiff_40_12	38128	14	0.7	200.695	3	2642.87	2642.87	0	276.22
Cardiff_40_13	121649	14	21.9	196.029	23	2723.1	2913.13	7	10800
Cardiff_40_14	247124	15	-	1716.28	1	2236.37	-	-	10800
Cardiff_40_15	33041	12	20.3	181.949	39	2606.74	2707.12	3.9	10800
Cardiff_40_16	153756	15	-	785.406	5	2927.97	-	-	10800
Cardiff_40_17	24463	13	3.2	278.477	23	2733.69	2781.87	1.8	10800
Cardiff_40_18	93744	14	-	224.516	21	2707.07	-	-	10800
Cardiff_40_19	10767	11	1.9	198.707	3	2784.05	2784.05	0	368.26
Cardiff_40_20	85341	14	-	214.132	23	2649.34	-	-	10800

Table 13: Detailed performance of the exact algorithm on 100-customer instances

Instance	$ \mathcal{A}^{\max} $	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
c101_100	18845	0.3	88.031	5	5298.76	5298.76	0	643.78
c102_100	18845	0.4	86.379	7	5298.36	5298.36	0	751.62
c103_100	18898	0.4	104.734	43	5296.46	5296.46	0	4008.0
c104_100	20064	1.4	141.282	121	5274.95	5339.06	1.2	10800
c105_100	18849	0.3	90.378	19	5298.76	5298.76	0	3249.86
c106_100	18844	0.4	90.421	9	5298.76	5298.76	0	1144.6
c107_100	18875	0.3	80.05	5	5298.76	5298.76	0	769.38
c108_100	18845	0.4	93.02	15	5295.36	5295.36	0	1712.0
c109_100	19018	0.4	125.412	49	5295.16	5295.16	0	5127.0

Table 14: Detailed performance of the exact algorithm on instances from (Gehring and Homberger, 1999)

Instance	$ \mathcal{A}^{\max} $	$\hat{\mathcal{K}}$	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
120_C1_2_1	14931	3	0.4	45.306	7	9090	9090	0	251.78
120_C1_2_2	14931	3	0.3	87.072	29	8920.5	8920.5	0	1827.92
120_C1_2_3	14931	3	1035	185.251	143	8815.06	-	-	10800
120_C1_2_4	14931	3	1046.7	163.474	153	8728.17	-	-	10800
120_C1_2_5	14931	3	0.5	58.236	19	8986.2	8986.2	0	1454.68
120_C1_2_6	14931	3	0.9	91.461	17	8988.4	8988.4	0	1899.7
120_C1_2_7	14931	3	0.5	67.444	29	8958.6	8958.6	0	2438.2
120_C1_2_8	14931	3	0.3	166.804	9	8817.9	8817.9	0	642.36
120_C1_2_9	14931	3	0.3	171.819	51	8790.62	8790.62	0	3394.73
120_C1_2_10	14931	3	1044.6	193.899	131	8741.16	-	-	10800
130_C1_2_1	17614	4	0.6	61.628	11	9325.92	9325.92	0	744.67
130_C1_2_2	17614	4	0.3	105.671	11	9141.2	9141.2	0	787.67
130_C1_2_3	17614	4	1005.5	147.702	105	9054.61	-	-	10800
130_C1_2_4	17617	4	1015.2	162.767	125	8973.53	-	-	10800
130_C1_2_5	17614	4	0.5	87.648	29	9209.62	9209.62	0	3294.54
130_C1_2_6	17614	4	0.8	119.465	109	9222.4	9222.4	0	9784.0
130_C1_2_7	17614	4	0.6	122.121	21	9197.92	9197.92	0	2199.14
130_C1_2_8	17614	4	0.7	148.914	153	9053.06	9091.7	0.4	10800
130_C1_2_9	17614	4	1008.9	177.902	107	9029.23	-	-	10800
130_C1_2_10	17614	4	1013.6	215.358	103	8985.44	-	-	10800
140_C1_2_1	20366	4	0.6	82.444	51	9970.42	9970.42	0	2840.77
140_C1_2_2	20366	4	0.5	135.393	129	9786.12	9817.38	0.3	10800
140_C1_2_3	20366	4	931.4	192.591	125	9710.4	-	-	10800
140_C1_2_4	20369	4	939	181.705	87	9632.6	-	-	10800
140_C1_2_5	20366	4	0.4	94.988	25	9848.82	9848.82	0	1981.57

Continued on next page

Table 14: Detailed performance of the exact algorithm on instances from (Gehring and Homberger, 1999)

Instance	$ \mathcal{A}^{\max} $	\hat{K}	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
140_C1_2_6	20366	4	1.1	121.619	89	9812.82	9891.3	0.8	10800
140_C1_2_7	20366	4	0.3	152.451	21	9836.32	9836.32	0	1867.67
140_C1_2_8	20366	4	0.6	193.816	103	9707.63	9753.78	0.5	10800
140_C1_2_9	20366	4	933	215.832	107	9683.85	-	-	10800
140_C1_2_10	20366	4	939.2	211.671	105	9636.41	-	-	10800
150_C1_2_1	23331	4	0.4	95.102	35	10699.3	10699.3	0	1539.33
150_C1_2_2	23331	4	0.6	202.898	103	10551.2	10599.8	0.5	10800
150_C1_2_3	23331	4	1.6	160.827	137	10453.6	10611.5	1.5	10800
150_C1_2_4	23334	4	860.4	221.244	91	10425.2	-	-	10800
150_C1_2_5	23331	4	0.6	107.043	123	10582.5	10628.2	0.4	10800
150_C1_2_6	23331	4	0.5	170.548	159	10561.4	10595.1	0.3	10800
150_C1_2_7	23331	4	0.5	131.694	113	10602.5	10602.5	0	5905.0
150_C1_2_8	23331	4	856.6	215.009	105	10468.6	-	-	10800
150_C1_2_9	23331	4	860.5	236.564	105	10416.2	-	-	10800
150_C1_2_10	23331	4	865	210.899	107	10381	-	-	10800

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_01	2	2	2	1	1795	83.27	3	3023.92	2931.88	3.14	108.18
Cardiff_40_01	2	3	3	1	1931	89.94	3	2931.88	2931.88	0.00	115.41
Cardiff_40_01	2	5	5	1	2491	111.19	1	2931.88	2931.88	0.00	112.07
Cardiff_40_01	2	5	8	1	5261	147.21	1	2931.88	2931.88	0.00	147.91
Cardiff_40_01	3	3	5	1	2366	138.05	1	2931.88	2931.88	0.00	138.87
Cardiff_40_01	3	5	5	1	2501	113.29	3	2931.88	2931.88	0.00	141.00
Cardiff_40_01	3	5	5	inf	2636	121.99	3	2931.88	2931.88	0.00	152.10
Cardiff_40_01	3	5	8	1	5283	175.87	3	2931.88	2931.88	0.00	231.41
Cardiff_40_01	3	5	8	inf	6601	213.12	3	2931.88	2931.88	0.00	276.06
Cardiff_40_01	3	8	8	1	5715	180.78	1	2931.88	2931.88	0.00	181.72
Cardiff_40_01	3	10	15	inf	16141	208.76	3	2931.88	2931.88	0.00	282.59
Cardiff_40_01	3	15	10	1	10714	197.94	3	2931.88	2931.88	0.00	273.71
Cardiff_40_01	3	15	15	inf	10714	190.00	3	2931.88	2931.88	0.00	252.57
Cardiff_40_02	2	2	2	1	1798	74.15	1	3258.43	3217.42	1.27	74.65
Cardiff_40_02	2	3	3	1	1940	76.30	1	3217.42	3217.42	0.00	76.65
Cardiff_40_02	2	5	5	1	2691	79.61	1	3217.42	3217.42	0.00	80.37
Cardiff_40_02	2	5	8	1	6439	119.73	3	3217.42	3217.42	0.00	368.83
Cardiff_40_02	3	3	5	1	2521	94.48	3	3217.42	3217.42	0.00	193.52
Cardiff_40_02	3	5	5	1	2707	99.99	1	3217.42	3217.42	0.00	100.56
Cardiff_40_02	3	5	5	inf	2831	73.55	3	3217.42	3217.42	0.00	163.43

Continued on next page

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root	Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_02	3	5	8	1	6743	110.46	7	3217.42	3217.42	0.00	1222.35	
Cardiff_40_02	3	5	8	inf	7965	117.88	13	3217.42	3217.42	0.00	1812.41	
Cardiff_40_02	3	8	8	1	7431	110.16	11	3217.42	3217.42	0.00	1928.13	
Cardiff_40_02	3	10	15	inf	94606	104.14	37	-	3217.42	-	10800.00	
Cardiff_40_02	3	15	10	1	15732	143.74	5	3217.42	3217.42	0.00	1813.37	
Cardiff_40_02	3	15	15	inf	44541	71.34	43	3217.42	3217.42	0.00	10800.00	
Cardiff_40_03	2	2	2	1	1798	64.23	3	2761.12	2682.5	2.93	92.10	
Cardiff_40_03	2	3	3	1	1932	83.51	1	2686.92	2682.5	0.16	84.03	
Cardiff_40_03	2	5	5	1	2573	101.58	3	2682.5	2682.5	0.00	193.17	
Cardiff_40_03	2	5	8	1	6071	150.94	3	2682.5	2682.5	0.00	277.87	
Cardiff_40_03	3	3	5	1	2441	83.80	3	2682.5	2682.5	0.00	138.85	
Cardiff_40_03	3	5	5	1	2605	107.42	3	2682.5	2682.5	0.00	186.08	
Cardiff_40_03	3	5	5	inf	2676	119.35	1	2682.5	2682.5	0.00	119.95	
Cardiff_40_03	3	5	8	1	6311	126.94	3	2682.5	2682.5	0.00	252.25	
Cardiff_40_03	3	5	8	inf	7550	132.35	7	2682.5	2682.5	0.00	447.35	
Cardiff_40_03	3	8	8	1	6983	141.43	5	2682.5	2682.5	0.00	470.14	
Cardiff_40_03	3	10	15	inf	32435	203.85	5	2682.5	2682.5	0.00	531.98	
Cardiff_40_03	3	15	10	1	14395	184.13	21	2682.5	2682.5	0.00	2524.46	
Cardiff_40_03	3	15	15	inf	20100	185.76	5	2682.5	2682.5	0.00	456.20	
Cardiff_40_04	2	2	2	1	1792	60.52	5	5259.18	5082.62	3.47	119.43	
Cardiff_40_04	2	3	3	1	1934	60.83	5	5239.48	5082.62	3.09	164.87	
Cardiff_40_04	2	5	5	1	2720	72.19	3	5239.48	5082.62	3.09	137.02	
Cardiff_40_04	2	5	8	1	7764	90.31	39	5245.75	5082.62	3.21	10800.00	
Cardiff_40_04	3	3	5	1	2573	79.86	7	5236.25	5082.62	3.02	224.70	
Cardiff_40_04	3	5	5	1	2781	69.25	7	5236.25	5082.62	3.02	196.73	
Cardiff_40_04	3	5	5	inf	2784	72.60	17	5236.25	5082.62	3.02	434.19	
Cardiff_40_04	3	5	8	1	8532	162.82	73	5142.45	5082.62	1.18	10800.00	
Cardiff_40_04	3	5	8	inf	8641	167.72	51	5178.7	5082.62	1.89	10800.00	
Cardiff_40_04	3	8	8	1	9603	115.39	49	-	5082.62	-	10800.00	
Cardiff_40_04	3	10	15	inf	169680	429.19	9	-	5082.62	-	10800.00	
Cardiff_40_04	3	15	10	1	26672	195.76	45	5082.62	5082.62	0.00	10800.00	
Cardiff_40_04	3	15	15	inf	146716	396.58	11	-	5082.62	-	10800.00	
Cardiff_40_05	2	2	2	1	1795	166.61	3	3812.85	3770.9	1.11	332.22	
Cardiff_40_05	2	3	3	1	1947	142.14	3	3812.85	3770.9	1.11	208.14	
Cardiff_40_05	2	5	5	1	2582	185.46	3	3770.9	3770.9	0.00	250.84	
Cardiff_40_05	2	5	8	1	5696	266.27	15	3770.9	3770.9	0.00	1436.28	
Cardiff_40_05	3	3	5	1	2453	207.88	3	3770.9	3770.9	0.00	273.77	
Cardiff_40_05	3	5	5	1	2609	213.88	3	3770.9	3770.9	0.00	269.74	
Cardiff_40_05	3	5	5	inf	2758	223.78	3	3770.9	3770.9	0.00	294.98	
Cardiff_40_05	3	5	8	1	5955	282.85	11	3770.9	3770.9	0.00	1134.08	
Cardiff_40_05	3	5	8	inf	7149	336.73	9	3770.9	3770.9	0.00	1313.44	
Cardiff_40_05	3	8	8	1	6492	300.18	13	3770.9	3770.9	0.00	1614.24	

Continued on next page

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root	Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_05	3	10	15	inf	25801	319.38	7	3770.9	3770.9	0.00	1614.57	
Cardiff_40_05	3	15	10	1	12799	397.57	5	3770.9	3770.9	0.00	1000.81	
Cardiff_40_05	3	15	15	inf	16776	239.74	3	3770.9	3770.9	0.00	674.01	
Cardiff_40_06	2	2	2	1	1800	108.36	1	3013.15	2909	3.58	108.80	
Cardiff_40_06	2	3	3	1	1939	111.47	1	2909.0	2909	0.00	112.15	
Cardiff_40_06	2	5	5	1	2671	167.08	3	2909.0	2909	0.00	263.32	
Cardiff_40_06	2	5	8	1	6008	193.26	3	2909.0	2909	0.00	331.11	
Cardiff_40_06	3	3	5	1	2501	157.41	1	2909.0	2909	0.00	158.25	
Cardiff_40_06	3	5	5	1	2680	177.21	3	2909.0	2909	0.00	292.64	
Cardiff_40_06	3	5	5	inf	2858	174.96	1	2909.0	2909	0.00	175.67	
Cardiff_40_06	3	5	8	1	6041	239.07	5	2909.0	2909	0.00	536.92	
Cardiff_40_06	3	5	8	inf	7529	235.25	3	2909.0	2909	0.00	387.91	
Cardiff_40_06	3	8	8	1	6590	256.48	3	2909.0	2909	0.00	503.01	
Cardiff_40_06	3	10	15	inf	26451	211.78	7	2909.0	2909	0.00	865.43	
Cardiff_40_06	3	15	10	1	13386	262.61	3	2909.0	2909	0.00	425.62	
Cardiff_40_06	3	15	15	inf	17176	175.53	5	2909.0	2909	0.00	591.90	
Cardiff_40_07	2	2	2	1	1796	77.83	1	2510.25	2491.27	0.76	78.45	
Cardiff_40_07	2	3	3	1	1947	81.88	1	2510.25	2491.27	0.76	82.69	
Cardiff_40_07	2	5	5	1	2756	96.54	3	2508.6	2491.27	0.70	220.03	
Cardiff_40_07	2	5	8	1	6981	141.71	7	2498.08	2491.27	0.27	711.42	
Cardiff_40_07	3	3	5	1	2579	115.86	3	2508.6	2491.27	0.70	239.77	
Cardiff_40_07	3	5	5	1	2781	112.91	3	2508.6	2491.27	0.70	243.98	
Cardiff_40_07	3	5	5	inf	2884	116.35	3	2508.6	2491.27	0.70	205.18	
Cardiff_40_07	3	5	8	1	7195	173.82	7	2498.08	2491.27	0.27	772.11	
Cardiff_40_07	3	5	8	inf	9005	154.60	3	2498.08	2491.27	0.27	362.34	
Cardiff_40_07	3	8	8	1	7704	170.55	3	2498.08	2491.27	0.27	394.30	
Cardiff_40_07	3	10	15	inf	144776	701.13	3	-	2491.27	-	10800.00	
Cardiff_40_07	3	15	10	1	18094	224.58	3	2491.27	2491.27	0.00	414.07	
Cardiff_40_07	3	15	15	inf	70597	375.00	9	-	2491.27	-	10800.00	
Cardiff_40_08	2	2	2	1	1798	67.13	5	3964.98	3834.02	3.42	288.31	
Cardiff_40_08	2	3	3	1	1932	70.08	7	3962.05	3834.02	3.34	263.64	
Cardiff_40_08	2	5	5	1	2590	100.03	3	3880.87	3834.02	1.22	194.73	
Cardiff_40_08	2	5	8	1	5238	143.00	1	3834.02	3834.02	0.00	143.70	
Cardiff_40_08	3	3	5	1	2444	138.10	3	3880.87	3834.02	1.22	223.55	
Cardiff_40_08	3	5	5	1	2605	114.78	3	3870.2	3834.02	0.94	205.67	
Cardiff_40_08	3	5	5	inf	2703	115.98	3	3870.2	3834.02	0.94	191.49	
Cardiff_40_08	3	5	8	1	5467	226.22	5	3834.02	3834.02	0.00	725.98	
Cardiff_40_08	3	5	8	inf	6622	235.22	5	3834.02	3834.02	0.00	683.52	
Cardiff_40_08	3	8	8	1	5922	217.93	5	3834.02	3834.02	0.00	751.23	
Cardiff_40_08	3	10	15	inf	9510	229.68	5	3834.02	3834.02	0.00	799.80	
Cardiff_40_08	3	15	10	1	6941	181.43	3	3834.02	3834.02	0.00	384.79	

Continued on next page

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root	Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_08	3	15	15	inf	6941	201.69		5	3834.02	3834.02	0.00	631.42
Cardiff_40_09	2	2	2	1	1799	148.26		3	4122.2	3668.2	12.38	323.28
Cardiff_40_09	2	3	3	1	1950	105.21		3	4114.32	3668.2	12.16	172.51
Cardiff_40_09	2	5	5	1	2698	186.22		5	3977.35	3668.2	8.43	469.85
Cardiff_40_09	2	5	8	1	6482	180.19		33	3726.48	3668.2	1.59	10800.00
Cardiff_40_09	3	3	5	1	2530	224.38		3	3955.0	3668.2	7.82	367.35
Cardiff_40_09	3	5	5	1	2717	222.18		3	3924.03	3668.2	6.97	334.68
Cardiff_40_09	3	5	5	inf	2830	236.50		3	3924.03	3668.2	6.97	369.09
Cardiff_40_09	3	5	8	1	6656	421.09		49	3668.2	3668.2	0.00	5115.00
Cardiff_40_09	3	5	8	inf	8011	446.94		49	3719.7	3668.2	1.40	10800.00
Cardiff_40_09	3	8	8	1	7350	323.74		31	3668.2	3668.2	0.00	10800.00
Cardiff_40_09	3	10	15	inf	80282	380.71		15	4639.5	3668.2	26.48	10800.00
Cardiff_40_09	3	15	10	1	15941	361.56		65	3668.2	3668.2	0.00	10800.00
Cardiff_40_09	3	15	15	inf	42829	362.02		21	3876.85	3668.2	5.69	10800.00
Cardiff_40_10	2	2	2	1	1800	76.35		1	2769.2	2692.08	2.86	76.72
Cardiff_40_10	2	3	3	1	1950	76.03		3	2769.2	2692.08	2.86	114.10
Cardiff_40_10	2	5	5	1	2669	85.22		1	2711.32	2692.08	0.71	86.05
Cardiff_40_10	2	5	8	1	5995	141.77		3	2711.32	2692.08	0.71	276.20
Cardiff_40_10	3	3	5	1	2515	99.54		1	2692.08	2692.08	0.00	100.29
Cardiff_40_10	3	5	5	1	2690	100.58		1	2692.08	2692.08	0.00	100.99
Cardiff_40_10	3	5	5	inf	2799	103.25		1	2692.08	2692.08	0.00	103.87
Cardiff_40_10	3	5	8	1	6222	151.79		1	2692.08	2692.08	0.00	152.67
Cardiff_40_10	3	5	8	inf	7334	173.19		1	2692.08	2692.08	0.00	174.24
Cardiff_40_10	3	8	8	1	6779	136.26		1	2692.08	2692.08	0.00	136.99
Cardiff_40_10	3	10	15	inf	55901	167.17		5	2692.08	2692.08	0.00	521.83
Cardiff_40_10	3	15	10	1	12351	156.87		1	2692.08	2692.08	0.00	157.60
Cardiff_40_10	3	15	15	inf	28248	210.33		1	2692.08	2692.08	0.00	211.15
Cardiff_40_11	2	2	2	1	1792	5.47		1	unfeasible	unfeasible	0.00	5.69
Cardiff_40_11	2	3	3	1	1921	4.57		1	unfeasible	unfeasible	0.00	4.79
Cardiff_40_11	2	5	5	1	2619	9.09		1	unfeasible	unfeasible	0.00	9.75
Cardiff_40_11	2	5	8	1	6236	10.38		1	unfeasible	unfeasible	0.00	10.86
Cardiff_40_11	3	3	5	1	2467	8.26		1	unfeasible	unfeasible	0.00	8.76
Cardiff_40_11	3	5	5	1	2635	8.60		1	unfeasible	unfeasible	0.00	9.02
Cardiff_40_11	3	5	5	inf	2774	8.18		1	unfeasible	unfeasible	0.00	8.48
Cardiff_40_11	3	5	8	1	6535	10.62		1	unfeasible	unfeasible	0.00	10.95
Cardiff_40_11	3	5	8	inf	8065	12.39		1	unfeasible	unfeasible	0.00	13.12
Cardiff_40_11	3	8	8	1	7188	13.01		1	unfeasible	unfeasible	0.00	13.37
Cardiff_40_11	3	10	15	inf	48327	32.05		1	unfeasible	unfeasible	0.00	33.12
Cardiff_40_11	3	15	10	1	15181	16.39		1	unfeasible	unfeasible	0.00	17.03
Cardiff_40_11	3	15	15	inf	27035	23.10		1	unfeasible	unfeasible	0.00	23.63
Cardiff_40_12	2	2	2	1	1792	54.70		3	2692.03	2642.87	1.86	72.94
Cardiff_40_12	2	3	3	1	1929	53.43		3	2642.87	2642.87	0.00	66.04

Continued on next page

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root	Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_12	2	5	5	1	2577	60.04	3	2642.87	2642.87	0.00	105.03	
Cardiff_40_12	2	5	8	1	5911	99.59	3	2642.87	2642.87	0.00	145.91	
Cardiff_40_12	3	3	5	1	2435	81.01	3	2642.87	2642.87	0.00	129.47	
Cardiff_40_12	3	5	5	1	2596	85.92	3	2642.87	2642.87	0.00	140.96	
Cardiff_40_12	3	5	5	inf	2720	87.54	3	2642.87	2642.87	0.00	142.77	
Cardiff_40_12	3	5	8	1	5911	98.25	3	2642.87	2642.87	0.00	138.24	
Cardiff_40_12	3	5	8	inf	7484	109.45	3	2642.87	2642.87	0.00	168.09	
Cardiff_40_12	3	8	8	1	6505	109.39	3	2642.87	2642.87	0.00	159.03	
Cardiff_40_12	3	10	15	inf	81967	173.31	3	2642.87	2642.87	0.00	251.61	
Cardiff_40_12	3	15	10	1	14174	108.45	3	2642.87	2642.87	0.00	165.07	
Cardiff_40_12	3	15	15	inf	35393	157.84	3	2642.87	2642.87	0.00	217.98	
Cardiff_40_13	2	2	2	1	1798	76.77	1	2999.0	2827.82	6.05	77.11	
Cardiff_40_13	2	3	3	1	1935	74.68	1	2999.0	2827.82	6.05	75.12	
Cardiff_40_13	2	5	5	1	2676	116.97	1	2883.55	2827.82	1.97	117.87	
Cardiff_40_13	2	5	8	1	6437	141.25	57	2864.38	2827.82	1.29	4819.00	
Cardiff_40_13	3	3	5	1	2528	141.22	3	2881.38	2827.82	1.89	254.77	
Cardiff_40_13	3	5	5	1	2717	101.87	1	2827.82	2827.82	0.00	102.28	
Cardiff_40_13	3	5	5	inf	2853	99.53	1	2827.82	2827.82	0.00	100.08	
Cardiff_40_13	3	5	8	1	6635	142.07	79	2827.82	2827.82	0.00	6390.00	
Cardiff_40_13	3	5	8	inf	8202	152.43	67	-	2827.82	-	10800.00	
Cardiff_40_13	3	8	8	1	7251	151.86	73	2827.82	2827.82	0.00	8276.00	
Cardiff_40_13	3	10	15	inf	121449	206.87	23	2913.13	2827.82	3.02	10800.00	
Cardiff_40_13	3	15	10	1	16815	208.34	113	2827.82	2827.82	0.00	8293.00	
Cardiff_40_13	3	15	15	inf	60077	156.43	41	-	2827.82	-	10800.00	
Cardiff_40_14	2	2	2	1	1796	75.74	15	3091.18	2910.9	6.19	960.53	
Cardiff_40_14	2	3	3	1	1929	93.03	45	3053.88	2910.9	4.91	2932.49	
Cardiff_40_14	2	5	5	1	2680	78.02	63	3028.5	2910.9	4.04	3480.14	
Cardiff_40_14	2	5	8	1	6624	117.48	63	3023.08	2910.9	3.85	10800.00	
Cardiff_40_14	3	3	5	1	2515	108.57	139	2957.28	2910.9	1.59	9388.00	
Cardiff_40_14	3	5	5	1	2705	112.64	55	2957.28	2910.9	1.59	3692.00	
Cardiff_40_14	3	5	5	inf	2819	120.63	39	2957.28	2910.9	1.59	2816.44	
Cardiff_40_14	3	5	8	1	6992	170.78	53	-	2910.9	-	10800.00	
Cardiff_40_14	3	5	8	inf	8269	192.80	63	-	2910.9	-	10800.00	
Cardiff_40_14	3	8	8	1	7750	187.90	81	-	2910.9	-	10800.00	
Cardiff_40_14	3	10	15	inf	246924	1796.16	1	-	2910.9	-	10800.00	
Cardiff_40_14	3	15	10	1	21082	102.81	113	2910.9	2910.9	0.00	7905.00	
Cardiff_40_14	3	15	15	inf	116234	1094.78	3	-	2910.9	-	10800.00	
Cardiff_40_15	2	2	2	1	1797	82.77	1	2760.08	2707.12	1.96	83.05	
Cardiff_40_15	2	3	3	1	1942	106.92	1	2760.08	2707.12	1.96	107.20	
Cardiff_40_15	2	5	5	1	2643	125.59	3	2707.12	2707.12	0.00	205.99	
Cardiff_40_15	2	5	8	1	6574	174.10	3	2707.12	2707.12	0.00	318.79	
Cardiff_40_15	3	3	5	1	2506	118.92	3	2707.12	2707.12	0.00	186.76	

Continued on next page

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root	Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_15	3	5	5	1	2680	134.02	3	3	2707.12	2707.12	0.00	201.04
Cardiff_40_15	3	5	5	inf	2730	132.68	3	3	2707.12	2707.12	0.00	198.58
Cardiff_40_15	3	5	8	1	6721	172.12	3	3	2707.12	2707.12	0.00	330.60
Cardiff_40_15	3	5	8	inf	8005	180.66	9	3	2707.12	2707.12	0.00	966.44
Cardiff_40_15	3	8	8	1	7296	186.29	5	3	2707.12	2707.12	0.00	597.70
Cardiff_40_15	3	10	15	inf	37408	185.77	41	3	2716.77	2707.12	0.36	10800.00
Cardiff_40_15	3	15	10	1	15169	133.81	25	3	2707.12	2707.12	0.00	3442.63
Cardiff_40_15	3	15	15	inf	23134	200.91	35	3	2707.12	2707.12	0.00	6986.00
Cardiff_40_16	2	2	2	1	1796	152.34	3	3	4095.77	3353.42	22.14	325.30
Cardiff_40_16	2	3	3	1	1938	139.04	3	3	3378.82	3353.42	0.76	246.38
Cardiff_40_16	2	5	5	1	2634	157.76	3	3	3378.82	3353.42	0.76	290.33
Cardiff_40_16	2	5	8	1	6861	251.24	37	3	3415.72	3353.42	1.86	10800.00
Cardiff_40_16	3	3	5	1	2483	223.02	3	3	3378.82	3353.42	0.76	387.75
Cardiff_40_16	3	5	5	1	2660	146.76	3	3	3353.42	3353.42	0.00	271.65
Cardiff_40_16	3	5	5	inf	2771	161.23	3	3	3353.42	3353.42	0.00	291.54
Cardiff_40_16	3	5	8	1	7072	278.11	49	3	3353.42	3353.42	0.00	10800.00
Cardiff_40_16	3	5	8	inf	8688	351.24	27	3	3366.97	3353.42	0.40	10800.00
Cardiff_40_16	3	8	8	1	7898	296.23	61	3	3410.52	3353.42	1.70	10800.00
Cardiff_40_16	3	10	15	inf	152985	835.01	5	-	-	3353.42	-	10800.00
Cardiff_40_16	3	15	10	1	19971	335.19	15	3	3366.97	3353.42	0.40	10800.00
Cardiff_40_16	3	15	15	inf	65872	480.78	9	-	-	3353.42	-	10800.00
Cardiff_40_17	2	2	2	1	1798	92.36	3	3	2831.3	2749.83	2.96	115.20
Cardiff_40_17	2	3	3	1	1939	87.32	1	3	2777.07	2749.83	0.99	87.80
Cardiff_40_17	2	5	5	1	2624	128.11	1	3	2749.83	2749.83	0.00	129.20
Cardiff_40_17	2	5	8	1	5241	189.10	5	3	2749.83	2749.83	0.00	676.60
Cardiff_40_17	3	3	5	1	2474	162.52	3	3	2749.83	2749.83	0.00	235.68
Cardiff_40_17	3	5	5	1	2642	160.60	1	3	2749.83	2749.83	0.00	161.46
Cardiff_40_17	3	5	5	inf	2738	164.62	1	3	2749.83	2749.83	0.00	165.30
Cardiff_40_17	3	5	8	1	5350	225.94	3	3	2749.83	2749.83	0.00	364.92
Cardiff_40_17	3	5	8	inf	6334	204.08	7	3	2749.83	2749.83	0.00	928.89
Cardiff_40_17	3	8	8	1	5787	191.46	3	3	2749.83	2749.83	0.00	307.75
Cardiff_40_17	3	10	15	inf	24600	280.43	33	-	-	2749.83	-	10800.00
Cardiff_40_17	3	15	10	1	9307	233.03	7	3	2749.83	2749.83	0.00	1051.50
Cardiff_40_17	3	15	15	inf	16770	256.00	9	3	2749.83	2749.83	0.00	4918.00
Cardiff_40_18	2	2	2	1	1787	120.56	3	3	2898.75	2888.65	0.35	264.26
Cardiff_40_18	2	3	3	1	1915	82.22	3	3	2888.65	2888.65	0.00	146.25
Cardiff_40_18	2	5	5	1	2579	88.51	5	3	2888.65	2888.65	0.00	268.48
Cardiff_40_18	2	5	8	1	6586	110.40	57	3	2941.17	2888.65	1.82	10800.00
Cardiff_40_18	3	3	5	1	2435	120.02	5	3	2888.65	2888.65	0.00	378.09
Cardiff_40_18	3	5	5	1	2593	109.83	5	3	2888.65	2888.65	0.00	355.63
Cardiff_40_18	3	5	5	inf	2711	117.91	3	3	2888.65	2888.65	0.00	190.47
Cardiff_40_18	3	5	8	1	6763	155.29	75	3	2888.65	2888.65	0.00	9097.00

Continued on next page

Table 15: Detailed performance of the heuristic

Instance	D_h	w_h	k_h	p_h	$ \mathcal{A}^{\max} $	Root	Time	Nodes	PB	BKS	Gap	Time (s)
Cardiff_40_18	3	5	8	inf	8311	165.19	37	2888.65	2888.65	0.00	4932.00	
Cardiff_40_18	3	8	8	1	7454	153.60	57	2902.2	2888.65	0.47	10800.00	
Cardiff_40_18	3	10	15	inf	93353	233.75	21	-	2888.65	-	10800.00	
Cardiff_40_18	3	15	10	1	17614	94.82	93	2888.65	2888.65	0.00	10800.00	
Cardiff_40_18	3	15	15	inf	48958	156.16	29	-	2888.65	-	10800.00	
Cardiff_40_19	2	2	2	1	1800	100.68	3	2807.1	2784.05	0.83	206.02	
Cardiff_40_19	2	3	3	1	1956	125.08	3	2807.1	2784.05	0.83	272.73	
Cardiff_40_19	2	5	5	1	2690	181.37	3	2784.05	2784.05	0.00	345.11	
Cardiff_40_19	2	5	8	1	5057	147.83	3	2784.05	2784.05	0.00	312.69	
Cardiff_40_19	3	3	5	1	2541	174.93	3	2784.05	2784.05	0.00	298.16	
Cardiff_40_19	3	5	5	1	2710	172.47	3	2784.05	2784.05	0.00	306.01	
Cardiff_40_19	3	5	5	inf	2801	156.48	7	2784.05	2784.05	0.00	557.54	
Cardiff_40_19	3	5	8	1	5075	158.20	3	2784.05	2784.05	0.00	261.90	
Cardiff_40_19	3	5	8	inf	6009	167.29	3	2784.05	2784.05	0.00	297.71	
Cardiff_40_19	3	8	8	1	5412	169.81	11	2784.05	2784.05	0.00	924.35	
Cardiff_40_19	3	10	15	inf	11376	209.09	3	2784.05	2784.05	0.00	374.94	
Cardiff_40_19	3	15	10	1	7577	211.69	7	2784.05	2784.05	0.00	650.65	
Cardiff_40_19	3	15	15	inf	8484	189.95	3	2784.05	2784.05	0.00	345.95	
Cardiff_40_20	2	2	2	1	1798	104.94	41	3167.3	3083.32	2.72	4142.00	
Cardiff_40_20	2	3	3	1	1937	94.93	53	3167.3	3083.32	2.72	4879.00	
Cardiff_40_20	2	5	5	1	2668	104.20	3	3083.32	3083.32	0.00	249.27	
Cardiff_40_20	2	5	8	1	7541	126.64	51	3083.32	3083.32	0.00	10800.00	
Cardiff_40_20	3	3	5	1	2503	152.96	3	3083.32	3083.32	0.00	295.26	
Cardiff_40_20	3	5	5	1	2685	142.13	3	3083.32	3083.32	0.00	315.32	
Cardiff_40_20	3	5	5	inf	2798	138.74	3	3083.32	3083.32	0.00	283.32	
Cardiff_40_20	3	5	8	1	7844	162.42	71	3217.35	3083.32	4.35	10800.00	
Cardiff_40_20	3	5	8	inf	9585	183.54	51	3102.13	3083.32	0.61	10800.00	
Cardiff_40_20	3	8	8	1	8665	198.26	47	3109.15	3083.32	0.84	10800.00	
Cardiff_40_20	3	10	15	inf	89884	223.75	23	-	3083.32	-	10800.00	
Cardiff_40_20	3	15	10	1	20544	273.60	55	3089.5	3083.32	0.20	10800.00	
Cardiff_40_20	3	15	15	inf	55292	162.79	31	-	3083.32	-	10800.00	

Table 16: Detailed experiments on the impact of the ARCC

Instance	ARCC	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
120_C1_2_01	yes	0	58.04	1	9090	9090	0	58.94
120_C1_2_01	no	0	43.98	1	9090	9090	0	44.63
120_C1_2_02	yes	0	162.93	1	8920.5	8920.5	0	163.87
120_C1_2_02	no	0	145.72	1	8920.5	8920.5	0	146.54
120_C1_2_03	yes	0.2	250.16	5	8843.52	8843.52	0	428.45

Continued on next page

Table 16: Detailed experiments on the impact of the ARCC

Instance	ARCC	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
120_C1_2_03	no	0.3	185.19	9	8843.52	8843.52	0	433.68
120_C1_2_04	yes	0.7	240.19	187	8740.64	8784.92	0.51	10800
120_C1_2_04	no	0.6	165.93	241	8740	8784.92	0.51	10800
120_C1_2_05	yes	0	240.02	1	8986.2	8986.2	0	241.07
120_C1_2_05	no	0	237.6	1	8986.2	8986.2	0	238.7
120_C1_2_06	yes	0.3	395.59	9	8988.4	8988.4	0	847.64
120_C1_2_06	no	0.3	341.83	9	8988.4	8988.4	0	908.45
120_C1_2_07	yes	0.1	210.22	3	8958.6	8958.6	0	375.54
120_C1_2_07	no	0.1	242.94	3	8958.6	8958.6	0	419.64
120_C1_2_08	yes	0.1	229.57	11	8817.9	8817.9	0	375.16
120_C1_2_08	no	0.1	175.54	15	8817.9	8817.9	0	529.05
120_C1_2_09	yes	0	281.36	1	8790.62	8790.62	0	282.26
120_C1_2_09	no	0	189.73	1	8790.62	8790.62	0	190.56
120_C1_2_10	yes	0.7	264.75	97	8740.41	8794.92	0.62	10800
120_C1_2_10	no	0.7	155.31	125	8737.92	8794.92	0.65	10800
130_C1_2_01	yes	0	193.32	1	9325.92	9325.92	0	194.05
130_C1_2_01	no	0	188.75	1	9325.92	9325.92	0	189.49
130_C1_2_02	yes	0	201.23	1	9141.2	9141.2	0	202.12
130_C1_2_02	no	0	169.74	1	9141.2	9141.2	0	170.74
130_C1_2_03	yes	0.7	198.25	241	9056.65	9119	0.69	10800
130_C1_2_03	no	0.7	199.17	167	9059.79	9119	0.65	10800
130_C1_2_04	yes	0.8	254.91	223	8993.56	9056.21	0.70	10800
130_C1_2_04	no	0.8	176.29	101	8989.81	9056.21	0.74	10800
130_C1_2_05	yes	0	264.44	1	9209.62	9209.62	0	265.45
130_C1_2_05	no	0	214.59	1	9209.62	9209.62	0	215.36
130_C1_2_06	yes	0.3	336.13	33	9222.4	9222.4	0	2152.1
130_C1_2_06	no	0.3	403.74	15	9222.4	9222.4	0	1633.03
130_C1_2_07	yes	0	687.49	3	9197.92	9197.92	0	1027.5
130_C1_2_07	no	0	771.28	3	9197.92	9197.92	0	1236.02
130_C1_2_08	yes	0.4	375.96	141	9082.79	9091.7	0.10	10800
130_C1_2_08	no	0.3	432.57	137	9066.88	9091.7	0.27	10800
130_C1_2_09	yes	0.6	261.74	119	9028.84	9077.5	0.54	10800
130_C1_2_09	no	0.6	236.97	129	9029.86	9077.5	0.53	10800
130_C1_2_10	yes	0.8	277.84	119	8992	9060.5	0.76	10800
130_C1_2_10	no	0.8	193.69	121	8990.42	9060.5	0.78	10800
140_C1_2_01	yes	0	268.45	1	9970.42	9970.42	0	269.33
140_C1_2_01	no	0	224.69	3	9970.42	9970.42	0	259.88
140_C1_2_02	yes	0.4	323.74	53	9817.38	9817.38	0	3780.0
140_C1_2_02	no	0.4	241.88	53	9817.38	9817.38	0	3656.0
140_C1_2_03	yes	0.8	271.7	115	9722.55	9784.7	0.64	10800
140_C1_2_03	no	0.8	174.26	119	9715.29	9784.7	0.71	10800

Continued on next page

Table 16: Detailed experiments on the impact of the ARCC

Instance	ARCC	Root Gap	Root Time	Nodes	DB	PB	Gap Time (s)
140_C1_2_04	yes	1.4	288.78	143	9637.97	9762.48	1.29 10800
140_C1_2_04	no	1.4	220.5	119	9637.72	9762.48	1.29 10800
140_C1_2_05	yes	0	262.62	1	9848.82	9848.82	0 263.61
140_C1_2_05	no	0	219.09	1	9848.82	9848.82	0 220.18
140_C1_2_06	yes	0.1	638.89	3	9853.5	9853.5	0 881.09
140_C1_2_06	no	0.1	598.12	3	9853.5	9853.5	0 809.91
140_C1_2_07	yes	0	325.88	3	9836.32	9836.32	0 363.95
140_C1_2_07	no	0	229.26	3	9836.32	9836.32	0 304.98
140_C1_2_08	yes	0.4	374.68	165	9726.07	9754.28	0.29 10800
140_C1_2_08	no	0.5	274.15	129	9716.74	9754.28	0.39 10800
140_C1_2_09	yes	0.8	271.95	157	9684.21	9754	0.72 10800
140_C1_2_09	no	0.9	258.01	181	9690.01	9754	0.66 10800
140_C1_2_10	yes	2.9	296.33	113	9639.97	9900	2.70 10800
140_C1_2_10	no	2.9	185.31	115	9638.62	9900	2.71 10800

Table 17: Detailed experiments on the impact of the SPR

Instance	SPR	$ A^{\max} $	Root Gap	Root Time	Nodes	DB	PB	Gap Time (s)
Cardiff_40_01	yes	13034	0	268.41	1	2931.88	2931.88	0 269.49
Cardiff_40_01	no	13174	0	297.93	1	2931.88	2931.88	0 298.89
Cardiff_40_02	yes	95883	20.1	88.12	45	2827.85	3217.43	13.8 10800
Cardiff_40_02	no	97648	20.1	117.31	29	2774.49	3217.42	16 10800
Cardiff_40_03	yes	30318	0	400.03	1	2682.5	2682.5	0 401.32
Cardiff_40_03	no	30721	9.1	264.48	3	2682.5	2682.5	0 542.77
Cardiff_40_04	yes	169680	20.2	317.5	9	4282.36	5082.63	18.7 10800
Cardiff_40_04	no	178022	20.2	478.01	7	4281.08	5082.63	18.7 10800
Cardiff_40_05	yes	25801	1.6	543.24	27	3770.9	3770.9	0 6574.0
Cardiff_40_05	no	26586	1.6	603.97	23	3770.9	3770.9	0 4977.0
Cardiff_40_06	yes	26451	0	491.63	1	2909	2909	0 492.73
Cardiff_40_06	no	27449	0	559.3	1	2909	2909	0 560.32
Cardiff_40_07	yes	89523	19.8	696.68	3	2093.42	2491.28	19 10800
Cardiff_40_07	no	91461	19.8	1008.99	1	2080.22	2491.28	19.8 10800
Cardiff_40_08	yes	9510	0	215.16	1	3834.02	3834.02	0 215.85
Cardiff_40_08	no	9671	0	258.93	1	3834.02	3834.02	0 259.65
Cardiff_40_09	yes	81734	12.7	1084.47	9	3569.6	3668.21	2.8 10800
Cardiff_40_09	no	84161	13.1	1156.82	11	3541.86	3668.21	3.6 10800
Cardiff_40_10	yes	55913	0	220.62	1	2692.08	2692.08	0 221.98
Cardiff_40_10	no	59456	0	297.74	1	2692.08	2692.08	0 299.62
Cardiff_40_11	yes	48327	0	5.94	1	unfeasible	unfeasible	0 6.59

Continued on next page

Table 17: Detailed experiments on the impact of the SPR

Instance	SPR	$ A^{\max} $	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
Cardiff_40_11	no	51524	0	6.89	1	unfeasible	unfeasible	0	8.0
Cardiff_40_12	yes	38128	0	280.93	1	2642.87	2642.87	0	282.13
Cardiff_40_12	no	38637	0	351.3	1	2642.87	2642.87	0	353.38
Cardiff_40_13	yes	121649	18.4	199.97	17	2742.72	2827.83	3.1	10800
Cardiff_40_13	no	123340	18.4	274.22	15	2742.19	2827.83	3.1	10800
Cardiff_40_14	yes	247124	30.2	1794.42	1	2236.37	2910.91	30.2	10800
Cardiff_40_14	no	260625	30.2	3466.11	1	2236.37	2910.91	30.2	10800
Cardiff_40_15	yes	33041	19.9	281.4	37	2549.95	2707.12	6.2	10800
Cardiff_40_15	no	33301	20	351.13	35	2643.72	2707.13	2.4	10800
Cardiff_40_16	yes	153756	18	844.32	3	2904.8	3353.43	15.4	10800
Cardiff_40_16	no	156619	19.1	1134.11	3	2881.14	3353.43	16.4	10800
Cardiff_40_17	yes	24463	1.8	678.22	13	2735.28	2749.84	0.5	10800
Cardiff_40_17	no	24917	1.9	916.07	13	2722.56	2749.84	1	10800
Cardiff_40_18	yes	93744	31	213.52	21	2676.43	2888.66	7.9	10800
Cardiff_40_18	no	95960	31	291.94	15	2509.3	2888.66	15.1	10800
Cardiff_40_19	yes	10767	0	264.74	1	2784.05	2784.05	0	265.26
Cardiff_40_19	no	10810	0	262.37	1	2784.05	2784.05	0	263.17
Cardiff_40_20	yes	85341	27.2	219.7	21	2586.44	3083.33	19.2	10800
Cardiff_40_20	no	87042	27.2	322.25	13	2594.19	3083.33	18.9	10800
c100_01	yes	18845	0	55.84	1	5298.76	5298.76	0	56.75
c100_01	no	18845	0	57.8	1	5298.76	5298.76	0	58.64
c100_02	yes	18845	0	88.01	1	5298.36	5298.36	0	88.81
c100_02	no	18845	0	84.7	1	5298.36	5298.36	0	85.49
c100_03	yes	18898	0	139.25	1	5296.46	5296.46	0	140.3
c100_03	no	18898	0	150.06	1	5296.46	5296.46	0	150.89
c100_04	yes	20064	0.5	111.65	39	5294.66	5294.66	0	1042.88
c100_04	no	20064	0.5	99.7	45	5294.66	5294.66	0	1629.27
c100_05	yes	18849	0	56.7	1	5298.76	5298.76	0	57.6
c100_05	no	18849	0	57.66	1	5298.76	5298.76	0	58.43
c100_06	yes	18844	0	80.37	1	5298.76	5298.76	0	81.23
c100_06	no	18844	0	78.6	1	5298.76	5298.76	0	79.32
c100_07	yes	18875	0	65.13	1	5298.76	5298.76	0	66.0
c100_07	no	18875	0	66.53	1	5298.76	5298.76	0	67.33
c100_08	yes	18845	0	107.34	1	5295.36	5295.36	0	108.15
c100_08	no	18845	0	114.34	1	5295.36	5295.36	0	115.17
c100_09	yes	19018	0	218.49	1	5295.16	5295.16	0	219.57
c100_09	no	19018	0	226.06	1	5295.16	5295.16	0	227.08

Table 18: Detailed experiments on the impact of the restricted master heuristics

Instance	RMH	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
Cardiff_40_01	yes	0.3	214.13	3	2931.88	2931.88	0.0	306.84
Cardiff_40_01	no	0.5	234.17	5	2931.88	2931.88	0.0	447.26
Cardiff_40_02	yes	20.1	92.8	37	3044.72	3217.42	5.67	10800.0
Cardiff_40_02	no	21.1	99.25	77	2986.6	3242.98	8.58	10800.0
Cardiff_40_03	yes	2.7	168.23	3	2682.5	2682.5	0.0	351.02
Cardiff_40_03	no	14.6	114.83	21	2682.5	2682.5	0.0	1287.88
Cardiff_40_04	yes	inf	401.23	9	4292.31	-	-	10800.0
Cardiff_40_04	no	9.2	453.32	57	4297.88	4620.65	7.51	10800.0
Cardiff_40_05	yes	1.7	329.68	11	3770.9	3770.9	0.0	2141.71
Cardiff_40_05	no	1.8	378.99	15	3770.9	3770.9	0.0	1415.53
Cardiff_40_06	yes	12.5	203.15	19	2909.0	2909.0	0.0	1633.04
Cardiff_40_06	no	14.6	214.06	17	2909.0	2909.0	0.0	1067.8
Cardiff_40_07	yes	inf	674.04	3	2093.42	-	-	10800.0
Cardiff_40_07	no	21.9	741.64	53	2297.3	2535.67	10.38	10800.0
Cardiff_40_08	yes	0.6	191.29	3	3834.02	3834.02	0.0	358.6
Cardiff_40_08	no	0.5	247.14	5	3834.02	3834.02	0.0	509.05
Cardiff_40_09	yes	35.1	381.09	11	3580.1	4484.18	25.25	10800.0
Cardiff_40_09	no	17.5	338.15	73	3580.5	3707.23	3.54	10800.0
Cardiff_40_10	yes	1.3	171.29	3	2692.08	2692.08	0.0	573.32
Cardiff_40_10	no	1.9	184.87	5	2692.08	2692.08	0.0	441.81
Cardiff_40_11	yes	0.0	29.34	1	unfeasible	unfeasible	0.0	30.12
Cardiff_40_11	no	0.0	32.62	1	unfeasible	unfeasible	0.0	33.43
Cardiff_40_12	yes	0.7	200.69	3	2642.87	2642.87	0.0	276.22
Cardiff_40_12	no	0.8	200.06	3	2642.87	2642.87	0.0	307.18
Cardiff_40_13	yes	21.9	196.03	23	2723.1	2913.13	6.98	10800.0
Cardiff_40_13	no	inf	211.4	81	2722.68	-	-	10800.0
Cardiff_40_14	yes	inf	1716.28	1	2236.37	-	-	10800.0
Cardiff_40_14	no	inf	1914.77	13	2435.77	-	-	10800.0
Cardiff_40_15	yes	20.3	181.95	39	2606.74	2707.12	3.85	10800.0
Cardiff_40_15	no	22.9	157.89	53	2707.12	2707.12	0.0	3757.0
Cardiff_40_16	yes	inf	785.41	5	2927.97	-	-	10800.0
Cardiff_40_16	no	19.8	863.33	61	3188.12	3372.57	5.79	10800.0
Cardiff_40_17	yes	3.2	278.48	23	2733.69	2781.87	1.76	10800.0
Cardiff_40_17	no	2.1	291.8	57	2749.83	2749.83	0.0	4976.0
Cardiff_40_18	yes	inf	224.52	21	2707.07	-	-	10800.0
Cardiff_40_18	no	33.0	254.4	63	2731.72	2932.45	7.35	10800.0
Cardiff_40_19	yes	1.9	198.71	3	2784.05	2784.05	0.0	368.26
Cardiff_40_19	no	1.7	227.32	17	2784.05	2784.05	0.0	989.55
Cardiff_40_20	yes	inf	214.13	23	2649.34	-	-	10800.0
Cardiff_40_20	no	32.1	237.58	103	2617.56	3203.98	22.4	10800.0

Continued on next page

Table 18: Detailed experiments on the impact of the restricted master heuristics

Instance	RMH	Root Gap	Root Time	Nodes	DB	PB	Gap	Time (s)
c100_01	yes	0.3	88.03	5	5298.76	5298.76	0.0	643.78
c100_01	no	0.3	70.29	33	5298.76	5298.76	0.0	3785.0
c100_02	yes	0.4	86.38	7	5298.36	5298.36	0.0	751.62
c100_02	no	0.3	80.16	63	5298.36	5298.36	0.0	2802.46
c100_03	yes	0.4	104.73	43	5296.46	5296.46	0.0	4008.0
c100_03	no	0.4	111.98	81	5296.46	5296.46	0.0	3877.0
c100_04	yes	1.4	141.28	121	5274.95	5339.06	1.22	10800.0
c100_04	no	inf	151.2	121	5274.14	-	-	10800.0
c100_05	yes	0.3	90.38	19	5298.76	5298.76	0.0	3249.86
c100_05	no	0.3	93.02	61	5298.76	5298.76	0.0	5342.0
c100_06	yes	0.4	90.42	9	5298.76	5298.76	0.0	1144.6
c100_06	no	0.3	89.64	23	5298.76	5298.76	0.0	1840.12
c100_07	yes	0.3	80.05	5	5298.76	5298.76	0.0	769.38
c100_07	no	0.3	87.01	37	5298.76	5298.76	0.0	3516.1
c100_08	yes	0.4	93.02	15	5295.36	5295.36	0.0	1712.0
c100_08	no	0.4	113.6	25	5295.36	5295.36	0.0	1823.05
c100_09	yes	0.4	125.41	49	5295.16	5295.16	0.0	5127.0
c100_09	no	0.6	126.27	69	5295.16	5295.16	0.0	3874.0
120_C1_2_01	yes	0.4	45.31	7	9090.0	9090.0	0.0	251.78
120_C1_2_01	no	0.4	43.36	7	9090.0	9090.0	0.0	279.21
120_C1_2_02	yes	0.3	87.07	29	8920.5	8920.5	0.0	1827.92
120_C1_2_02	no	0.3	93.16	83	8920.5	8920.5	0.0	2895.06
120_C1_2_03	yes	inf	185.25	143	8815.06	-	-	10800.0
120_C1_2_03	no	0.7	158.94	247	8812.02	8868.12	0.64	10800.0
120_C1_2_04	yes	inf	163.47	153	8728.17	-	-	10800.0
120_C1_2_04	no	inf	155.55	229	8718.23	-	-	10800.0
120_C1_2_05	yes	0.5	58.24	19	8986.2	8986.2	0.0	1454.68
120_C1_2_05	no	0.5	65.29	21	8986.2	8986.2	0.0	1552.51
120_C1_2_06	yes	0.9	91.46	17	8988.4	8988.4	0.0	1899.7
120_C1_2_06	no	0.9	97.45	99	8988.4	8988.4	0.0	4892.0
120_C1_2_07	yes	0.5	67.44	29	8958.6	8958.6	0.0	2438.2
120_C1_2_07	no	0.4	84.31	29	8958.6	8958.6	0.0	1845.62
120_C1_2_08	yes	0.3	166.8	9	8817.9	8817.9	0.0	642.36
120_C1_2_08	no	0.3	153.67	135	8817.9	8817.9	0.0	6501.0
120_C1_2_09	yes	0.3	171.82	51	8790.62	8790.62	0.0	3394.73
120_C1_2_09	no	inf	164.03	231	8771.15	-	-	10800.0
120_C1_2_10	yes	inf	193.9	131	8741.16	-	-	10800.0
120_C1_2_10	no	1.0	191.08	243	8729.87	8808.82	0.9	10800.0

References

- Achterberg, T., 2007. Constraint Integer Programming. Ph.D. thesis. Technische Universität Berlin. URL: <https://depositonce.tu-berlin.de/items/9f46a10e-2f7b-4dea-8e27-9cae07de5258>, doi:doi:10.14279/depositonce-1634.
- Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science* 52, 965–981. URL: <https://pubsonline.informs.org/doi/10.1287/trsc.2017.0791>, doi:doi:10.1287/trsc.2017.0791.
- Amazon, 2023. Amazon is launching ultra-fast drone deliveries in Italy, the UK, and a third location in the U.S. <https://www.aboutamazon.com/news/operations/amazon-prime-air-drone-delivery-updates>. URL: <https://www.aboutamazon.com/news/operations/amazon-prime-air-drone-delivery-updates>. [Accessed 27-10-2023].
- Baldacci, R., Christofides, N., Mingozzi, A., 2008. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming* 115, 351–385. URL: <https://doi.org/10.1007/s10107-007-0178-5>, doi:doi:10.1007/s10107-007-0178-5.
- Baldacci, R., Mingozzi, A., Roberti, R., 2011. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research* 59, 1269–1283. URL: <http://www.jstor.org/stable/41316029>.
- Błażewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J., 2007. Handbook on scheduling: from theory to applications. Springer Science & Business Media.
- Bouman, P., Agatz, N., Schmidt, M., 2018. Dynamic programming approaches for the traveling salesman problem with drone. *Networks* 72, 528–542. URL: <https://onlinelibrary.wiley.com/doi/10.1002/net.21864>, doi:doi:10.1002/net.21864.
- Chen, C., Demir, E., Huang, Y., 2021. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research* 294, 1164–1180. URL: <https://linkinghub.elsevier.com/retrieve/pii/S037722172100120X>, doi:doi:10.1016/j.ejor.2021.02.027.
- Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research* 123, 105004. URL: <https://www.sciencedirect.com/science/article/pii/S0305054820301210>, doi:doi:10.1016/j.cor.2020.105004.

- Di Puglia Pugliese, L., Guerriero, F., 2017. Last-Mile Deliveries by Using Drones and Classical Vehicles, in: Sforza, A., Sterle, C. (Eds.), *Optimization and Decision Science: Methodologies and Applications*, Springer International Publishing, Cham. pp. 557–565. URL: https://link.springer.com/chapter/10.1007/978-3-319-67308-0_56, doi:doi:10.1007/978-3-319-67308-0_56.
- Di Puglia Pugliese, L., Guerriero, F., Macrina, G., 2020. Using drones for parcels delivery process. *Procedia Manufacturing* 42, 488–497. URL: <https://www.sciencedirect.com/science/article/pii/S2351978920305928>.
- Dorling, K., Heinrichs, J., Messier, G.G., Magierowski, S., 2016. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47, 70–85. URL: <https://ieeexplore.ieee.org/abstract/document/7513397>.
- DPDgroup, 2016. Innovation for delivering difficult-to-reach areas. URL: <https://www.dpd.com/fr/en/sending-parcels/our-innovative-services/drone-delivery/>.
- Dukkanci, O., Kara, B.Y., Bektaş, T., 2021. Minimizing energy and cost in range-limited drone deliveries with speed optimization. *Transportation Research Part C: Emerging Technologies* 125, 102985. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0968090X21000206>, doi:doi:10.1016/j.trc.2021.102985.
- Figliozzi, M.A., 2017. Lifecycle modeling and assessment of unmanned aerial vehicles (drones) co2e emissions. *Transportation Research Part D: Transport and Environment* 57, 251–261.
- Garg, V., Niranjana, S., Prybutok, V., Pohlen, T., Gligor, D., 2023. Drones in last-mile delivery: A systematic review on Efficiency, Accessibility, and Sustainability. *Transportation Research Part D: Transport and Environment* 123, 103831. URL: <https://linkinghub.elsevier.com/retrieve/pii/S1361920923002286>, doi:doi:10.1016/j.trd.2023.103831.
- Gehring, H., Homberger, J., 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows, in: *Proceedings of EURO-GEN99*, Citeseer. pp. 57–64. URL: <https://www.semanticscholar.org/paper/A-Parallel-Hybrid-Evolutionary-Metaheuristic-for-Gehring/e41b80d075429403f569980d278f6d4f91da2594>.
- Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey, in: *Annals of discrete mathematics*. Elsevier. volume 5, pp. 287–326. URL: <https://www.sciencedirect.com/science/article/abs/pii/S016750600870356X>.
- Jackson, J.R., 1955. Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project .

- Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research* 56, 497–511. URL: <https://doi.org/10.1287/opre.1070.0449>, doi:doi:10.1287/opre.1070.0449.
- Kang, M., Lee, C., 2021. An Exact Algorithm for Heterogeneous Drone-Truck Routing Problem. *Transportation Science* 55, 1088–1112. URL: <http://pubsonline.informs.org/doi/10.1287/trsc.2021.1055>, doi:doi:10.1287/trsc.2021.1055.
- Karak, A., Abdelghany, K., 2019. The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies* 102, 427–449. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0968090X18312932>, doi:doi:10.1016/j.trc.2019.03.021.
- Kitjacharoenchai, P., Min, B.C., Lee, S., 2020. Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics* 225, 107598. URL: <https://www.sciencedirect.com/science/article/pii/S0925527319304335>, doi:doi:10.1016/j.ijpe.2019.107598.
- Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J.M.A., Brunese, P.A., 2019. Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering* 129, 14–30. URL: <https://www.sciencedirect.com/science/article/pii/S0360835219300245>, doi:doi:10.1016/j.cie.2019.01.020.
- Laporte, G., Nobert, Y., 1983. A branch and bound algorithm for the capacitated vehicle routing problem. *Operations-Research-Spektrum* 5, 77–85. URL: <https://doi.org/10.1007/BF01720015>, doi:doi:10.1007/BF01720015.
- Leishman, G.J., 2006. Principles of helicopter aerodynamics with CD extra. Cambridge university press.
- Lin, C.A., Shah, K., Mauntel, L.C.C., Shah, S.A., 2018. Drone delivery of medications: Review of the landscape and legal considerations. *The Bulletin of the American Society of Hospital Pharmacists* 75, 153–158. URL: <https://academic.oup.com/ajhp/article-abstract/75/3/153/5102071>.
- Lysgaard, J., Letchford, A.N., Eglese, R.W., 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical programming* 100, 423–445. URL: <https://link.springer.com/article/10.1007/s10107-003-0481-8>.
- Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laporte, G., 2020. A literature review: Drone-aided routing:. *Transportation Research Part C: Emerging Technologies* 120, 102762. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X20306744>, doi:doi:10.1016/j.trc.2020.102762.

- Meng, S., Guo, X., Li, D., Liu, G., 2023. The multi-visit drone routing problem for pickup and delivery services. *Transportation Research Part E: Logistics and Transportation Review* 169, 102990. URL: <https://www.sciencedirect.com/science/article/pii/S136654522003672>, doi:doi:10.1016/j.tre.2022.102990.
- Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies* 54, 86–109. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0968090X15000844>, doi:doi:10.1016/j.trc.2015.03.005.
- Murray, C.C., Raj, R., 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies* 110, 368–398. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X19302505>, doi:doi:10.1016/j.trc.2019.11.003.
- Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., 2017a. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation* 9, 61–100. URL: <https://link.springer.com/article/10.1007/s12532-016-0108-8>.
- Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., Santos, H., 2017b. Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters* 45, 206–209.
- Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2018. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing* 30, 339–360. URL: <https://doi.org/10.1287/ijoc.2017.0784>, doi:doi:10.1287/ijoc.2017.0784.
- Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2020. A generic exact solver for vehicle routing and related problems. *Mathematical Programming* 183, 483–523. URL: <https://link.springer.com/10.1007/s10107-020-01523-z>, doi:doi:10.1007/s10107-020-01523-z.
- Poikonen, S., Wang, X., Golden, B., 2017. The vehicle routing problem with drones: Extended models and connections. *Networks* 70, 34–43. URL: <https://onlinelibrary.wiley.com/doi/10.1002/net.21746>, doi:doi:10.1002/net.21746.
- Post, S., 2023. Swiss post drone transport in the healthcare sector. URL: <https://www.post.ch/en/about-us/innovation/innovations-in-development/drones>.
- Roca-Riu, M., Menendez, M., 2019. Logistic deliveries with drones: State of the art of practice and research, in: 19th Swiss Transport Research Conference (STRC 2019), STRC. URL: <https://www.research-collection.ethz.ch/handle/20.500.11850/342823>.

- Sacramento, D., Pisinger, D., Ropke, S., 2019. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies* 102, 289–315. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0968090X18303218>, doi:doi:10.1016/j.trc.2019.02.018.
- Sadykov, R., Uchoa, E., Pessoa, A., 2021. A Bucket Graph-Based Labeling Algorithm with Application to Vehicle Routing. *Transportation Science* 55, 4–28. URL: <http://pubsonline.informs.org/doi/10.1287/trsc.2020.0985>, doi:doi:10.1287/trsc.2020.0985.
- Sadykov, R., Vanderbeck, F., 2021. BaPCod - a generic branch-and-price code. Technical Report. Inria Bordeaux Sud-Ouest. URL: <https://inria.hal.science/hal-03340548>.
- Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., Uchoa, E., 2019. Primal heuristics for branch and price: The assets of diving methods. *INFORMS Journal on Computing* 31, 251–267. URL: <https://doi.org/10.1287/ijoc.2018.0822>, doi:doi:10.1287/ijoc.2018.0822.
- Schermer, D., Moeini, M., Wendt, O., 2019. A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research* 109, 134–158. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0305054819301042>, doi:doi:10.1016/j.cor.2019.04.021.
- Sluijk, N., Florio, A.M., Kinable, J., Dellaert, N., Van Woensel, T., 2023. Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research* 304, 865–886. URL: <https://www.sciencedirect.com/science/article/pii/S0377221722001278>.
- Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research* 35, 254–265. URL: <https://pubsonline.informs.org/doi/abs/10.1287/opre.35.2.254>.
- Stankovic, J.A., Spuri, M., Ramamritham, K., Buttazzo, G.C., 1998. Fundamentals of EDF Scheduling, in: *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. Springer US, Boston, MA, pp. 27–65. URL: https://doi.org/10.1007/978-1-4615-5535-3_3, doi:doi:10.1007/978-1-4615-5535-3_3.
- Tamke, F., Buscher, U., 2021. A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological* 144, 174–203. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0191261520304410>, doi:doi:10.1016/j.trb.2020.11.011.
- Tamke, F., Buscher, U., 2023. The vehicle routing problem with drones and drone speed selection. *Computers & Operations Research* 152, 106112. URL: <https://www.sciencedirect.com/science/article/pii/S0305054822003422>, doi:doi:10.1016/j.cor.2022.106112.

- Ulmer, M.W., Thomas, B.W., 2018. Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks* 72, 475–505. URL: <https://onlinelibrary.wiley.com/doi/10.1002/net.21855>, doi:doi:10.1002/net.21855.
- Wang, D., 2015. The economics of drone delivery. URL: <https://www.flexport.com/blog/drone-delivery-economics/>.
- Wang, X., Poikonen, S., Golden, B., 2017. The vehicle routing problem with drones: several worst-case results. *Optimization Letters* 11, 679–697. URL: <http://link.springer.com/10.1007/s11590-016-1035-3>, doi:doi:10.1007/s11590-016-1035-3.
- Wang, Z., Sheu, J.B., 2019. Vehicle routing problem with drones. *Transportation Research Part B: Methodological* 122, 350–364. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0191261518307884>, doi:doi:10.1016/j.trb.2019.03.005.
- Yang, Y., Yan, C., Cao, Y., Roberti, R., 2023. Planning robust drone-truck delivery routes under road traffic uncertainty. *European Journal of Operational Research* 309, 1145–1160. URL: <https://www.sciencedirect.com/science/article/pii/S0377221723001649>, doi:doi:10.1016/j.ejor.2023.02.031.
- Yin, Y., Qing, L., Wang, D., Cheng, T.C.E., Ignatius, J., 2024. Exact solution method for vehicle-and-drone cooperative delivery routing of blood products. *Computers & Operations Research* 164, 106559. URL: <https://www.sciencedirect.com/science/article/pii/S0305054824000315>, doi:doi:10.1016/j.cor.2024.106559.
- Zhou, H., Qin, H., Cheng, C., Rousseau, L.M., 2023. An exact algorithm for the two-echelon vehicle routing problem with drones. *Transportation Research Part B: Methodological* 168, 124–150. URL: <https://www.sciencedirect.com/science/article/pii/S0191261523000024>, doi:doi:10.1016/j.trb.2023.01.002.
- Zhu, X., 2019. Segmenting the public’s risk beliefs about drone delivery: A belief system approach. *Telematics and Informatics* 40, 27–40. URL: <https://www.sciencedirect.com/science/article/pii/S0736585319302692>.