Received: 10 September 2018 | Accepted: 10 September 2018 | Published on: 24 October 2018

DOI: 10.1002/net.21858

SPECIAL ISSUE ARTICLE

WILEY

Drone arc routing problems

James F. Campbell[1] | **Ángel Corberán**[2] | **Isaac Plana**[3] | **José M. Sanchis**[4]

[1]Department of Marketing, College of Business Administration, University of Missouri–St. Louis, St. Louis, Missouri
[2]Departament d'Estadística i Investigació Operativa, Universitat de València, Valencia Spain
[3]Departament de Matemátiques per a l'Economia i l'Empresa, Universitat de València, Valencia Spain
[4]Departamento de Matemática Aplicada, Instituto Universitario de Matemática Pura y Aplicada, Universidad Politécnica de Valencia, Valencia, Spain

Correspondence
José M. Sanchis, Universidad Politécnica de Valencia, Valencia, Spain.
Email: jmsanchis@mat.upv.es

Funding information
This research was supported by the Ministerio de Economía y Competitividad and FEDER - European Regional Development Fund, MTM2015-68097-P.

Abstract

In this article, we present some drone arc routing problems (Drone ARPs) and study their relation with well-known postman ARPs. Applications for Drone ARPs include traffic monitoring by flying over roadways, infrastructure inspection such as by flying along power transmission lines, pipelines or fences, and surveillance along linear features such as coastlines or territorial borders. Unlike the postmen in traditional ARPs, drones can travel directly between any two points in the plane without following the edges of the network. As a consequence, a drone route may service only part of an edge, with multiple routes being used to cover the entire edge. Thus the Drone ARPs are continuous optimization problems with an infinite number of feasible solutions. In order to solve them as a discrete optimization problem, we approximate each curve in the plane by a polygonal chain, thus allowing the vehicle to enter and leave each curve only at the points of the polygonal chain. If the capacity of the vehicles is unlimited, the resulting problem is a rural postman problem (RPP). We propose an algorithm that iteratively solves RPP instances with an increasing number of points of the polygonal chain and present results on several sets of instances. We also briefly discuss the case in which the drones have limited capacity and several drones are needed.

KEYWORDS

cutting path problems, drones, rural postman problem

1 | INTRODUCTION

Drones, or UAVs (unmanned aerial vehicles), provide new opportunities for improving logistics in a variety of settings. We use the term drone for the unmanned vehicle and focus on aerial vehicles, though this research may apply to sea surface and subsurface unmanned vessels as well [24]. The use of drones in routing applications can lead to improved service from higher speeds, reduced costs, and safety improvements by removing human operators. Drone ARPs extend traditional ARPs to allow vehicles (drones) to travel directly between any two points on the network (or directly between a depot and any point on the network). Thus, the drone travels along the network while servicing edges, but it can also travel off the network. Examples of Drone ARPs include traffic monitoring by flying over roadways, infrastructure inspection such as by flying along power transmission lines, pipelines or fences [21], and surveillance along linear features, as with coastlines or territorial borders. Unlike ground based vehicles that are limited to follow local ground infrastructure such as roadways or paths (including ground drones; e.g., www.starship.xyz), aerial drones allow more direct travel and can easily fly across areas without roads, over bodies of water, and so on. Note that in some cases there may be three dimensional (3D) ARPs, as for inspecting a set of wind turbines [34] or undersea infrastructure [20, 22, 36].

Because the drone can travel off the network, it may service only part of an edge, with multiple routes being used to cover the entire edge. A further consequence is that the physical network (e.g., of roadways or pipelines) cannot be modeled simply

as an abstract graph with the appropriate connections and edge lengths, as that does not properly capture the distance (or cost) to travel between any two points in the network not following the edges. This added flexibility in Drone ARPs requires new modeling approaches.

Drone delivery problems, such as for home delivery of small consumer goods, healthcare deliveries of vaccines, blood or defibrillators and disaster response tend to be modeled as extensions of well-studied node routing or facility location problems (e.g., [1–3, 19, 26, 31, 32]). On the other hand, drone use for inspection sensing and surveillance along linear features can be seen as extensions of well-studied ARPs (i.e., the Chinese postman problem [CPP] and the rural postman problem [RPP]). Otto et al. [30] review academic research on optimization approaches for a variety of routing problems for the commercial use of drones.

The idea in Drone ARPs is to use drones to optimize the coverage or "servicing" (imaging, inspection, surveillance, etc.) of designated edges in a network, denoted "required" edges, where the network could be pipelines, roads, railroads, fences, boundaries, shorelines, power transmission lines, infrastructure, and so on. We refer to the traversal of an edge requiring coverage as servicing the edge, and we assume that traversal of the edge in either direction satisfies the servicing. Drones may also traverse edges not required for coverage and may travel directly between two points of the network. We assume the drone can change directions instantly to match the edge connections in the network. Note that, in practice, the drone flight dynamics and travel motion depend on the type of drone (e.g., rotor copter vs fixed wing), where for example fixed wing drones have a minimum turning radius described by a Dubins path [13]. Futhermore, some sensors may require being parallel to the ground, thus limiting coverage while the drone is turning at an angle. Our models are most reflective of the flight dynamics for rotor copter drones.

In this research, we consider a set of homogeneous drones that must jointly service a set of edges of a network. Drones start and end at a specified vertex, the depot, and have a route length limit (or time) that applies to the total route (as from the battery or fuel capacity), whether servicing an edge on the network or traveling directly off the network. Applications where there is a capacity for servicing the edges, as for limited data storage from a sensor, is a subject of future research. Also, note that in some real-world settings, the usage of the drone capacity may depend on the activity, as servicing an edge may involve more intensive use of the battery (to record data or to send data back) than traveling off the network when not servicing, and the capacity usage rate may depend on the speed of travel. However, as a first step, we assume the capacity is a linear function of the distance traveled, and thus the drone capacity is captured in the route length limit.

There is some research on arc routing for robots or other unmanned vehicles, but much is really no different than having regular vehicles. Of interest here is research that combines travel on a network to service a subset of edges (typical arc routing) with travel direct between two points on the network with the Euclidean metric (for the drone). Although there is a rapidly growing literature on node routing problems with drones, only a few papers dealing with ARPs with aerial drones have been published. Oh et al. [27, 28] explore a road network search problem using drones where a set of unconnected road segments need to be searched. This is solved with a greedy insertion heuristic that models drone travel distance between the road components as a Dubins path, as for a fixed wing drone. Dille and Singh [10] also model drone travel using Dubins paths for a setting where the drone has a sensor with a radius of coverage. The authors convert the road (edge) covering problem to a TSP by discretizing the road network into a set of coverage points designed so that visiting each point will ensure that all parts of the road network are within the sensor range of a drone. Results were compared to the method in Oh et al. [28] for simulated urban, suburban and rural networks.

Chow [4] proposes a deterministic multi-period arc-inventory routing problem (AIRP) for deploying drones for traffic monitoring. The demand in the inventory routing problem models the need for drone monitoring based on data such as traffic conditions. The stochastic dynamic AIRP is also considered and solved with an approximate dynamic programming algorithm. Results for small instances with 7 and 11 links show the benefits over static myopic policies. Li et al. [23] also consider inventory routing in combination with the capacitated ARP (CARP) for road traffic monitoring. Over-monitoring of an edge is viewed as inventory and is penalized. A mixed integer programing model is presented and solved with CPLEX for up to 12 nodes and 40 links. Larger problems with up to 15 nodes and 50 links are solved with a local branching algorithm. A relevant paper with ground drones (robots) is Easton and Burdisck [14], which solves an RPP to route small ground robots for inspecting the boundaries (edges) of a set of nonoverlapping objects in two-dimensions (turbine blades). The boundaries to be inspected define edges of a graph that must be traversed by the robots to ensure that all boundaries are inspected. The robots are effectively allowed to travel directly between the boundaries of different objects and can inspect only part of an object's boundary. This is somewhat analogous to having drones travel directly between two points of a network and having a drone service only part of an edge. The problem is modeled as an RPP with k robots and solved with a construction heuristic. Two other references that use ground drones in ARPs are Sipahioglu et al. [35] and Yazici et al. [37]. These works consider a network that covers an indoor region that includes various obstacles, and construct routes heuristically for robots of limited capacity, where there are different capacity usage rates for servicing or not servicing an edge.

ARPs with a single drone also share some characteristics with the problems of generating paths for laser cutting machines or drawing plotters. Dewil et al. [9] classifies cutting path problems into six types, and the one which most resembles the

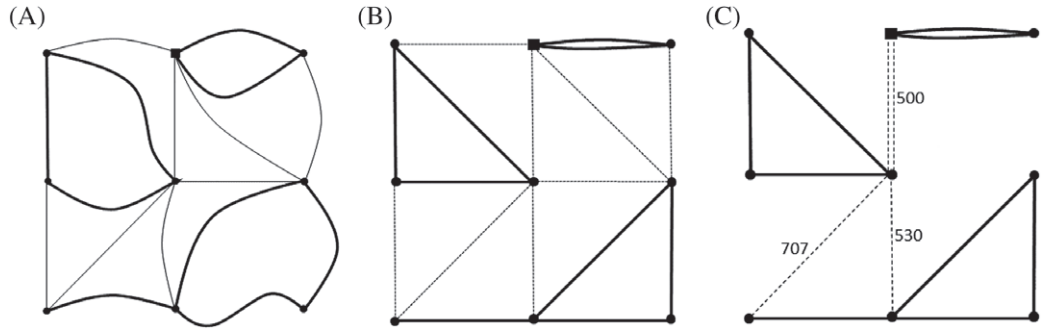

FIGURE 1 A rural postman problem instance and its optimal solution

problem studied in this paper is the Intermittent Cutting Problem (ICP), in which the object can be cut in sections and there is no restriction on the points that can be used for entry or exit. The only two works reported in [9] related to the ICP are those by Garfinkel and Webb [16], which will be discussed later, and Rodrigues and Soeiro [33], which proposes a memetic algorithm for the RPP and applies it to the solution of a cutting path problem appearing in an industrial application.

The remainder of the paper is organized as follows. Section 2 summarizes some results about postman ARPs. In Section 3, we present a Drone ARP with a single vehicle, discuss the differences with respect to the traditional postman problems and propose an algorithm for solving it to optimality. Computational experiments are reported in Section 4. The version with several vehicles is introduced in Section 5, and, finally, some conclusions are presented in Section 6.

2 | ARC ROUTING PROBLEMS

Given a network (roadways, electricity lines, pipelines, etc.) and a set of lines (straight or curved lines) that are required to be covered (serviced), each one with an associated cost, ARPs consist of finding a tour (closed walk starting and ending at the depot) covering all the required lines with total minimum cost. Figure 1A shows such a network, where the square represents the depot and the thick lines the required ones.

If the required set of lines to be covered is the whole set of lines in the network, the problem is known as the *CPP*. This problem was proposed by the Chinese mathematician Guan [18], who aimed to design the route for a postman (hence the name of Chinese postman), although its origins go back to the famous Königsberg bridges problem [15]. Orloff [29] defined the *RPP,* as a generalization of the CPP in which only a subset of the lines in the network has to be serviced. If we consider that the service is performed by a fleet of vehicles with limited capacity, we have the *CARP*, introduced by Golden and Wong [17]. The CARP aims to find a set of minimum cost routes that jointly cover all the required edges while each particular route satisfies the capacity of the vehicle. For a comprehensive treatment of this area, see the books edited by Dror [12] and by Corberán and Laporte [6], as well as the recent annotated bibliography by Mourão and Pinto [25].

In the following, we will call these postman ARPs. Postman ARPs are usually solved in a graph, where each line is represented by an edge or arc with an associated cost (Figure 1B). In addition to the required edges (drawn in bold lines), which must be traversed and serviced, the graph can have other nonrequired edges (drawn in thin lines), that can be traversed (deadheaded) to complete the tour. In these problems, it is assumed that:

- Each edge has to be completely traversed (starting at one of its end vertices and finishing at the other one), and
- From the final vertex of one edge to the initial vertex of another edge, the postman can only travel through the edges of the network.

In Figure 1C the optimal solution of the RPP instance in Figure 1B is represented. Its cost is the sum of the costs of the required edges (required cost) plus 2237 (deadheading cost). Note that in the RPP the sum of the costs of the required edges is a fixed value that is shared by all the feasible solutions. Therefore, in the RPP the goal is to minimize the deadheading cost.

3 | DRONES AND ARPs WITH A SINGLE VEHICLE

Let us consider in this section that the vehicle used to cover some lines of a given network is a drone with unlimited capacity and, therefore, one vehicle is enough to perform the service. Unlike postmen, whether they are walking or traveling with ground vehicles, drones can travel directly between any two points in the plane without following the edges of the network. In other words, when dealing with ARPs with drones, we can assume that:

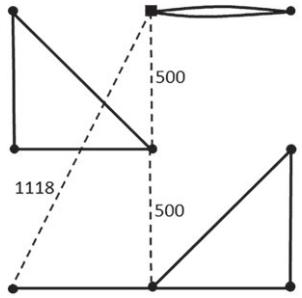

FIGURE 2 Drone optimal tour with Assumption 1

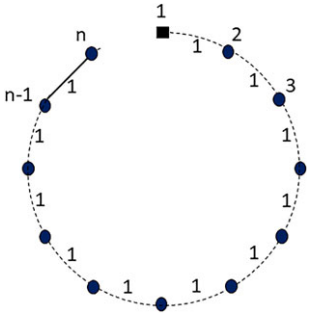

FIGURE 3 ARP instance used in the proof of Theorem 1 [Colour figure can be viewed at wileyonlinelibrary.com]

Assumption 1. After finishing the service of an edge at one of its vertices, the drone can travel directly to any vertex of another edge without following the edges of the network.

Assumption 1 implies that in ARPs with drones the set of nonrequired edges form a complete graph and the deadheading cost between any two points of the plane is given by the Euclidean distance. As a result, better solutions than for postman ARPs can be obtained. For example, Figure 2 shows, for the RPP instance presented in Figure 1B, the drone optimal tour with Assumption 1. This drone tour has a deadheading cost of 2118, which is less than the deadheading cost (2237) of the optimal RPP tour shown in Figure 1C. Note that the drone for Figure 2 travels directly between the central vertex and the vertex directly below it at cost 500, rather than following the nonrequired edge of cost 530 as in Figure 1C.

For Drone ARPs with Assumption 1, we have two theoretical results. Let $G = (V, E)$ be an undirected and connected graph. Let $E_R \subseteq E$ ($E_{NR} \subseteq E$) denote the set of the required (nonrequired) edges and let $G_R = (V_R, E_R)$ be the graph induced by the edges in E_R and the depot. We will assume that, as is usual in ARPs, the deadheading cost of a required edge is less than or equal to its service cost.

Theorem 1. *The ratio between the cost of the optimal postman tour and the cost of the optimal drone tour can be as large as desired.*

Proof. Consider the graph $G = (V, E)$ depicted in Figure 3, with set of vertices $\{1, 2, 3, \ldots, n\}$, nonrequired edges between consecutive nodes $(1, 2)$, $(2, 3)$, …, $(n-2, n-1)$, and one required edge $(n-1, n)$. If all the edges have cost 1, the cost of the optimal postman tour is $2(n-1)$, while the cost of the optimal drone tour is at most 4. Then the ratio is at least $\frac{2(n-1)}{4}$, which tends to infinity when n grows. ∎

Theorem 2. *If graph $G_R = (V_R, E_R)$ is connected, then the cost of the optimal postman tour is at most twice the cost of the optimal drone tour, and this value is attainable.*

Proof. Since the drone must traverse all the required edges, the cost of its optimal tour, C_D, will be at least the sum of the costs of the required edges, C_R. Given that G_R is connected, there is a feasible tour for the postman consisting of traversing each required edge twice, with total cost not greater than $2C_R$. Then the optimal tour for the postman has a cost $C_P \leq 2C_R \leq 2C_D$.

In order to see that the above ratio of 2 is attainable, consider the instance in Figure 4 with two required edges with both service and deadheading cost 1 and a common vertex, and the two other vertices located at a short distance of ϵ

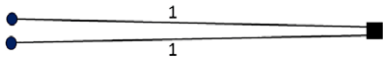

FIGURE 4 ARP instance used in the proof of Theorem 2 [Colour figure can be viewed at wileyonlinelibrary.com]

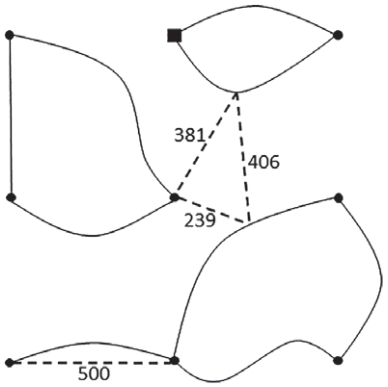

FIGURE 5 Optimal solutions with Assumptions 1 + 2

TABLE 1 Optimal deadheading costs of the RPP with different assumptions

RPP	2237
RPP with drones with Assumption 1	2118
RPP with drones with Assumptions 1 + 2	1526

from each other. It is easy to see that the cost of the optimal postman tour is 4 while the cost of the optimal drone tour is $2+\epsilon$, thus obtaining a ratio that tends to 2 as ϵ tends to 0. ■

Given that drones can travel directly between any two points in the plane (without following the edges of the network), in Drone ARPs, we can also assume that:

Assumption 2. Drones may start and end the service of an edge at any point of it.

Assumption 2 implies that the shape of the lines must now be taken into account. This is another difference with respect to postman ARPs, where the geometric shape of the edge is irrelevant, as long as its cost is given (since the vehicle can only enter or leave an edge from its endpoints). Assumption 2 can make the savings in the Drone ARPs solutions with respect to the postman ARP solutions even greater. For example, Figure 5 shows the drone optimal tour with Assumptions 1 and 2 for the RPP instance represented in Figure 1A . This tour has a deadheading cost of 1526.

As a summary, Table 1 shows the deadheading cost of the different optimal tours of the instance in Figure 1 when solved, respectively, as a "classical" RPP, and as a RPP with drones with the different Assumptions 1 and 2.

With Assumptions 1 and 2, we can define the Drone RPP as follows:

Definition 1. (Drone RPP) Given a set of lines, each one with an associated service cost, and a point called the depot, and assuming that the cost of deadheading between any two points is the Euclidean distance, find the minimum cost tour starting and ending at the depot that services all the given lines.

Note that Definition 1 is not restricted to a plane, though for clarity our illustrations will be planar. A simpler version of this problem was proposed in Garfinkel and Webb [16], who called it the crossing postman problem (XPP). As in the Drone RPP, in the XPP it is permitted to leave the edges of the network and travel from one edge to another at points other than the original vertices. Nevertheless, there is no mention in that paper of the possibility of the required edges being curved lines. Garfinkel and Webb proved that the XPP is NP-hard and mentioned that it could be applied to the problem of finding the shortest cycle for a machine that cuts polygonally shaped patterns from a material such as cloth or metal. They also presented a number of results that, for the RPP and XPP, allow certain nonrequired edges to be eliminated without sacrificing all optimal solutions. However, they did not develop any solution algorithm for the XPP.

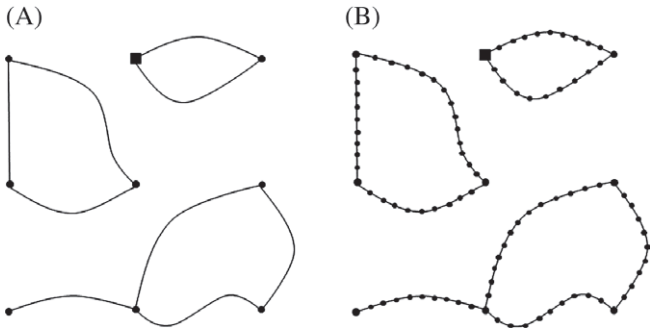

FIGURE 6 A Drone RPP instance

It is known that the RPP reduces to the CPP when the graph G_R is connected, and this is also true for the XPP [16]. It is easy to see that, in this case, the Drone RPP also reduces to the CPP in the sense that its optimal solution is found by obtaining a minimum cost matching on the odd-degree vertices. Therefore Assumption 2 has no effect on the solution of the Drone RPP when the required lines induce only one connected component.

The drawback of Definition 1 is the difficulty of modeling a curved line. In postman ARPs, the network is simply a set of edges, and each edge is represented by a pair of integers, (i, j), and a cost c_{ij}. This does not work for Drone ARPs.

In the real-world network, a Drone RPP instance is a family of lines (see Figure 6A). A curved line joining two points v, w can be approximated by a polygonal chain, formed by point v, an ordered set of *intermediate* points, and point w (see Figure 6B). The better defined we want the curve to be, the greater the number of intermediate points we will need. All these points are given by their coordinates (x_i, y_i). Each segment in a polygonal chain has associated a service cost, so that the sum of all the costs associated to all the segments of the polygonal chain equals the service cost of the original curve. Additionally, since the drone can travel directly between any two points of the plane, the deadheading costs are the Euclidean distances computed from the coordinates of the points. We define a Drone RPP instance as a family of polygonal chains as follows:

Definition 2. (Drone RPP instance) A Drone RPP instance contains a list of polygonal chains. For each chain we have the following data:

v, w, c_{vw}, n (the endpoints, service cost, number of intermediate points),
i_1, c_1 (intermediate point 1, service cost for the segment (v, i_1)),
i_2, c_2 (intermediate point 2, service cost for the segment (i_1, i_2)),
$\vdots$
i_n, c_n (intermediate point n, service cost for the segment (i_{n-1}, i_n)),
w, c_{n+1} (original endpoint w, service cost for the segment (i_n, w)),

with the costs satisfying $c_1 + c_2 + \cdots + c_n + c_{n+1} = c_{vw}$. A Drone RPP instance also contains a list with the coordinates of each point (the endpoints, the intermediate points and the depot).

3.1 | On the solution of the Drone RPP

The Drone RPP can be solved as an undirected RPP on a graph defined as follows. The set of vertices contains all the endpoints and all the intermediate points. The required edges are the segments in all the polygonal chains, with costs equal to the service cost of the segments. The nonrequired edges form a complete graph with costs equal to the Euclidean distances. For example, the Drone RPP instance in Figure 6 can be modeled as an RPP instance defined in the graph in Figure 6B, with $|V| = 98$, $|E_R| = 98$, and $|E_{NR}| = 4753$ (not drawn in the figure).

Strictly speaking, if the depot is not incident with a required edge, instead of an RPP instance, we would have a *general routing problem* (GRP) instance. The GRP is the problem of finding a minimum cost tour that visits a given subset of vertices and traverses a given subset of edges of a network. It contains the RPP and CPP as special cases. However, since the depot is the only possible isolated required vertex, for the sake of simplicity, we will continue to discuss RPP instances.

A Drone RPP instance contains all the points and segments in the polygonal chains, and can therefore be extremely large. For example, the last instance shown in Table 2, consisting of 84 polygonal chains with 20 intermediate points per chain on average, has 1680 vertices, 1681 required edges and may have up to 1 410 360 nonrequired edges (although some of them can be removed, as will be mentioned later). A standard RPP instance with these characteristics may be very hard to solve optimally.

TABLE 2 Characteristics of the "random" Drone RPP instances

Instance	Original vertices	Original lines	Total vertices	Segments	Potential NR edges	Final NR edges	*R*-connected components
DroneRPP561	22	18	364	360	66 066	30 895	5
DroneRPP562	23	22	440	439	96 580	48 328	3
DroneRPP581	34	25	508	499	128 778	47 430	9
DroneRPP582	34	35	701	702	245 350	119 486	2
DroneRPP5101	37	30	607	600	183 921	80 982	7
DroneRPP5102	44	49	977	982	476 776	177 005	4
DroneRPP661	26	21	425	420	90 100	37 221	6
DroneRPP662	29	26	523	520	136 503	56 251	4
DroneRPP681	34	28	564	558	158 766	71 544	7
DroneRPP682	42	45	897	900	401 856	103 762	3
DroneRPP6101	49	40	807	798	325 221	100 663	13
DroneRPP6102	52	58	1153	1159	664 128	228 080	6
DroneRPP771	38	30	607	599	183 921	72 969	9
DroneRPP772	39	49	970	980	469 965	17 721	2
DroneRPP791	35	45	910	900	413 595	140 671	11
DroneRPP792	58	58	1158	1158	669 903	145 436	8
DroneRPP7101	58	51	1027	1020	526 851	230 624	10
DroneRPP7102	59	65	1290	1296	831 405	92 868	5
DroneRPP881	55	46	932	923	433 846	150 720	12
DroneRPP882	59	61	1216	1218	738 720	254 922	4
DroneRPP891	61	53	1068	1060	569 778	179 358	12
DroneRPP892	59	61	1217	1219	739 936	168 723	6
DroneRPP8101	69	56	1129	1116	636 756	241 701	16
DroneRPP8102	68	78	1549	1559	1 198 926	150 073	6
DroneRPP991	64	56	1121	1113	627 760	208 699	10
DroneRPP992	71	79	1570	1578	1 231 665	285 715	6
DroneRPP9101	74	60	1208	1194	729 028	282 590	16
DroneRPP9102	81	78	1560	1557	1 216 020	368 756	15
DroneRPP10101	80	67	1356	1343	918 690	329 459	14
DroneRPP10102	83	84	1680	1681	1 410 360	554 314	9

However, due to its special structure, the Drone RPP instance may not be as difficult to solve as it might seem for the following reasons:

(a) Recall that the difficulty of solving the RPP depends strongly on the number of R-connected components (the connected components of graph $G_R = (V_R, E_R)$), and that the Drone RPP instance has the same number of R-connected components as the initial problem.

(b) The (potentially huge) number of nonrequired edges can be reduced as the following nonrequired edges can be removed:

- Nonrequired edges joining two vertices in the same R-connected component, except the edges joining two odd-degree vertices (see Lemma 3 by Garfinkel and Webb in [16]). The reason is as follows: Consider a feasible solution deadheading an edge (i, j), with vertices i, j in the same R-connected component and j being an even-degree vertex. This solution also deadheads an edge (j, k), for some vertex k, because j is an even-degree vertex. If we replace the traversals of (i, j) and (j, k) by the traversal of (i, k), we obtain a feasible solution with no greater cost. Note that edge (i, k) exists because E_{NR} induces a complete graph.
- Nonrequired edges parallel to required ones if they have the same cost (see Christofides et al. [5]).
- Nonrequired edges (i, j) for which there is a vertex k such that $c_{ij} = c_{ik} + c_{kj}$ (see Christofides et al. [5]). In fact, since the costs in Drone ARPs are real numbers, we remove the nonrequired edges (i, j) such that $c_{ij} \approx c_{ik} + c_{kj}$.

(c) In addition to the Drone RPP instance, we can generate smaller RPP instances by considering only a subset of intermediate points from each polygonal chain. These RPP instances, called RPP(k) in Section 3.2, are easier to solve and they provide upper bounds that can help to solve the Drone RPP instance. Furthermore, the feasible solutions of the small RPP instances are also feasible solutions for the Drone RPP instance.

3.2 | The global algorithm

Let us suppose we have a Drone RPP instance considered as a family of polygonal chains defined with a reasonable number of evenly spaced points (otherwise we could add such points). Given an integer $k \geq 0$, RPP(k) will denote the RPP instance formed by selecting, for each polygonal chain, k of its intermediate points. Note that RPP(k) corresponds to the case where drones are allowed to start and end the service of an edge either at its extreme points or at the selected k intermediate points. More specifically, RPP(k) is generated in the following way:

> **Definition 3.** (RPP(k) instance) Given a Drone RPP instance and given an integer $k \geq 0$, we call RPP(k) the RPP instance defined on a graph whose set of vertices contains, for each polygonal chain (v, w), the points v, w and k of its intermediate points as evenly spaced as possible These consecutive $k+2$ points define a new polygonal chain with $k+1$ segments. These new segments are the required edges, with a service cost equal to the sum of the service costs of the corresponding original segments. The nonrequired edges form a complete graph and their (deadheading) costs are computed from the coordinates of the selected vertices.

Instance RPP(0) (without intermediate points) represents the case in which drones are allowed to start and end the service of an edge only at its extreme points (Assumption 1). Instance RPP(1) consists of taking, for each polygonal chain, the two extreme points and the most centered intermediate point. It represents the case in which each original line is modeled with two segments that, jointly, have the service cost of the original line. Therefore, it will hopefully provide better solutions than RPP(0). The price to pay is that RPP(1) has, approximately, twice as many vertices and required edges as RPP(0) and many more nonrequired edges. Note that, as the vertex set of RPP(0) is included in the vertex set of RPP(1), each feasible solution for RPP(0) can be considered as a feasible solution for RPP(1) with the same cost (only replacing each required edge of RPP(0) by the two consecutive required edges of RPP(1), which in total have the same service cost).

In instance RPP(3), each line is represented by 4 segments defined by 5 points: the three points of RPP(1) -the two extreme points and a central intermediate point- plus an intermediate point as centered as possible between each extreme point and the central point. Again, it will hopefully provide better solutions than RPP(1). Note that a feasible solution for RPP(1) is a feasible solution for RPP(3), but not necessarily for RPP(2). This is the reason why the algorithm below jumps from RPP(1) to RPP(3), skipping RPP(2).

In general, given integers $k < k'$ such that the vertex set of RPP(k) is included in the vertex set of RPP(k'), each feasible solution for RPP(k) is a feasible solution for RPP(k') with the same cost.

The proposed algorithm works as follows:

> **Algorithm.**
>
> 1. Generate and solve RPP(0). Let z_0 be its optimal cost or the cost of the best feasible solution found.
> 2. Generate RPP(1) and solve it with z_0 as an upper bound. Let z_1 be its optimal cost or the cost of the best feasible solution found.
> 3. Generate RPP(3) and solve it with $\min\{z_0, z_1\}$ as an upper bound. Let z_3 be its optimal cost or the cost of the best feasible solution found.
> 4. Solve the Drone RPP instance with $\min\{z_0, z_1, z_3\}$ as an upper bound. If solved, the solution is optimal.
> 5. The best solution obtained (if any) is either the optimal solution (if obtained in (4)) or a feasible (maybe optimal) solution otherwise.

Note that the above scheme can be modified in order to consider additional steps related to the solution of RPP(k) instances for other values of k. For the instance sizes that we handle in this paper, RPP(3) represents a good balance between the computational effort and the quality of the obtained solutions.

The RPP(k) instances are solved using the branch-and-cut algorithm proposed in Corberán et al. [7] for the windy GRP (WGRP), which is a generalization of the RPP. This is a very sophisticated branch-and-cut algorithm that incorporates several families of valid inequalities such as connectivity, R-odd cut, K-C, Honeycomb, Path-Bridge and Zigzag inequalities. It is capable of solving to optimality WGRP instances with up to 1000 vertices, 4000 edges, and 500 R-connected components.

4 | COMPUTATIONAL EXPERIMENTS

In this section, we present the instances that have been used in these experiments and report the computational results obtained.

4.1 | Instance generation

In order to evaluate the performance of the solution method proposed, we have randomly generated some Drone RPP instances (families of polygonal chains, as defined previously) on a grid as follows. We first select values for the following parameters:

- n, m size of the original grid,
- p probability that an edge of the original grid is declared required,
- *nsplits* approximate number of segments into which an edge of average length is split,
- *curvature* measure of how curved the parabola replacing an edge is,
- *costfactor* ratio of the service costs with respect to the deadheading costs.

Then, we proceed as follows:

1. We generate an RPP instance on a grid with $n \times m$ points whose coordinates are multiples of 100. Initially, the graph contains all the vertical and horizontal edges, as well as some diagonals chosen at random. An edge is considered required with probability p. We randomly select the depot. The vertices, except for the depot, that are not incident with required edges are removed.
2. The coordinates (x_i, y_i) of the vertices are randomly perturbed (by adding a random value in $[-20, 20]$ to each coordinate) to prevent edges that are completely horizontal or vertical.
3. We compute the parameter *step* (the approximate length of a segment) as

$$step = \frac{average\{c_{ij} \ : \ (i,j) \in E_R\}}{nsplits}$$

4. For each edge $(i, j) \in E_R$,
   - Compute $nsplits(i,j) = \max\left\{int\left(\frac{c_{ij}}{step} + 0.5\right), 1\right\}$
   - Apply procedure *split* $(x_i, y_i, x_j, y_j, nsplits(i, j), curvature)$

The resulting Drone RPP instance has as many lines as required edges in the original RPP instance. A line corresponding to an edge (i, j) consists of a polygonal chain containing point i, the $nsplits(i, j) - 1$ intermediate points computed by the procedure *split*, and point j. The service cost of each segment of the polygonal chain is the Euclidean cost multiplied by *costfactor*. The service cost of the line (i, j) is equal to the sum of the service costs of all the segments that form the corresponding polygonal chain. The instance also includes the coordinates of all the vertices (original and intermediate).

The procedure *split*$(x_i, y_i, x_j, y_j, nsplits(i, j), curvature)$ generates the $nsplits(i, j) - 1$ intermediate points between (x_i, y_i) and (x_j, y_j) in the parabola that passes through points (x_i, y_i), (x_j, y_j), and a third point randomly selected using the parameter *curvature*. More specifically:

- If $x_i > x_j$, exchange the role of the points (x_i, y_i), (x_j, y_j).
- Compute $d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$.
- Consider the points of the plane $(x_1, y_1) = (0, 0)$ and $(x_2, y_2) = (d_{ij}, 0)$. Randomly select a third "intermediate" point (x_3, y_3):

$$x_3 \in [d_{ij}(1 - curvature)/2, d_{ij} - d_{ij}(1 - curvature)/2],$$
$$y_3 \in [-d_{ij} \times curvature/2, +d_{ij} \times curvature/2].$$

- Calculate the parabola $y = Ax^2 + Bx + C$ passing through the 3 points (x_1, y_1), (x_2, y_2) and (x_3, y_3): $A = y_3/(x_3(x_3 - d_{ij}))$, $B = -d_{ij}A$, $C = 0$.
- Compute the $nsplits(i, j) - 1$ intermediate points of the parabola $y = Ax^2 + Bx + C$: for each $k = 1, \ldots, nsplits(i, j) - 1$

$$x_k = k \times \frac{d_{ij}}{nsplits(i,j)},$$
$$y_k = Ax_k^2 + Bx_k + C$$

TABLE 3 Characteristics of the "even" Drone RPP instances

Instance	Original vertices	Original lines	Total vertices	Segments	Potential NR edges	Final NR edges	*R*-connected components
DroneRPP56	22	24	477	479	113 526	63 445	4
DroneRPP58	34	29	585	580	170 820	61 924	10
DroneRPP510	44	47	939	942	440 391	132 866	9
DroneRPP66	26	23	463	460	106 953	63 332	7
DroneRPP68	34	32	639	637	203 841	98 230	9
DroneRPP610	49	42	847	840	358 281	112 081	15
DroneRPP77	39	46	912	919	415 416	226 185	6
DroneRPP79	58	58	1134	1134	642 411	207 287	10
DroneRPP710	58	54	1087	1083	590 241	261 592	11
DroneRPP88	55	46	927	918	429 201	152 033	14
DroneRPP89	61	55	1100	1094	604 450	223 080	14
DroneRPP810	69	67	1343	1341	901 153	335 794	12
DroneRPP99	64	61	1217	1241	739 936	292 996	10
DroneRPP910	81	86	1715	1720	1 469 755	471 435	12
DroneRPP1010	83	92	1832	1841	1 677 196	674 725	11

- Translate and rotate the previous (x_k, y_k) points from the segment between the points (0, 0) and $(d_{ij}, 0)$ to the segment between the original points (x_i, y_i), (x_j, y_j). Let $t_0 = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$. For each point $k = 1, \ldots, nsplits(i,j) - 1$

$$r_k = \sqrt{x_k^2 + y_k^2},$$
$$t_k = \arctan\left(\frac{y_k}{x_k}\right),$$
$$x_k := x_1 + r_k \cos(t_0 + t_k),$$
$$y_k := y_1 + r_k \sin(t_0 + t_k).$$

With different values for $m, n \in \{5, 6, 7, 8, 9, 10\}$, $nsplits = 20$, and $costfactor = 1.5$, we have generated two sets of Drone RPP instances: one with $P = 0.3$ and $curvature = 0.5$ (version 1), and another with $p = 0.4$ and $curvature = 0.4$ (version 2). The characteristics of these instances are shown in Table 2, and can be found in http://www.uv.es/~corberan/instancias.htm. For example, the instance named "DroneRPP5102" was generated with $m = 5$, $n = 10$, and $p = 0.4$, $curvature = 0.4$ (version 2).

Table 2 shows, for each Drone RPP instance, the number of vertices and lines of the original instance, the total number of vertices (end nodes and intermediate nodes) and the number of segments. The number of nonrequired edges of the complete graph is given in the column "potential NR edges," while column "final NR edges" reports the number of nonrequired edges after applying the simplification procedure described in Section 3.1. Finally, the last column shows the number of *R*-connected components. Note that, with $nsplits = 20$, the size of the instances, especially the number of nonrequired edges, is huge.

In these randomly generated instances, the number of vertices incident with an odd number of required edges is high (approximately half of them), and, as can be seen in the computational results section, the optimal solutions obtained for the Drone RPP instances are similar to those obtained for the RPP(k) ones. Therefore, we have generated some more instances with fewer odd-degree vertices. The goal is to obtain instances with solutions that can be significantly different for the different RPP(k) and Drone RPP instances and which can be more difficult for our branch-and-cut algorithm. At the same time, we will try to make the graphs of these instances resemble the shapes that appear in cutting path problems. To achieve this, we have drawn some of the instances in Table 2 and modified them by adding and removing lines in order to reduce the number of odd vertices. The characteristics of these new "even" instances are shown in Table 3. An example of one of these instances, as well as its optimal solution, is shown in Figure 7.

4.2 | Computational results

Table 4 shows the computational results obtained for the 30 randomly generated instances described in Table 2. They have been obtained with a time limit of 1 hour for the WGRP branch-and-cut algorithm on a Intel Core i7 (Intel Corporation, Santa Clara, CA, USA) at 3.4 GHz with 32 GB RAM. Columns 2 and 3 report for each instance the cost of the optimal solution of RPP(0) and the computing time in seconds. For RPP(1), RPP(3), and Drone RPP, the table shows the improvement in percentage of their

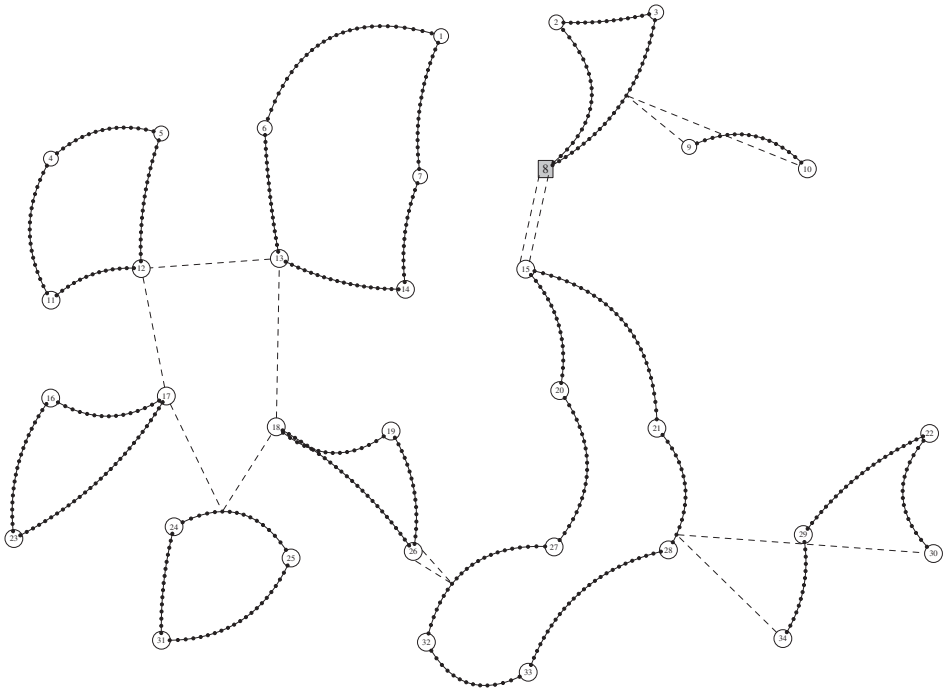

FIGURE 7 Optimal solution of instance DroneRPP68

optimal solutions with respect to the RPP(0) optimal solution and their corresponding computing times. For those instances that have not been solved optimally within the time limit, we have computed the difference in percentage between their lower bounds and the RPP(0) optimal solutions. These values, marked with an asterisk, represent an upper bound to the improvement that can be obtained by solving the Drone RPP with respect to the solution of RPP(0).

All the RPP(0), RPP(1), and RPP(3) instances have been easily solved to optimality. These instances provide very good feasible solutions for the Drone RPP that are, in many cases, optimal. Although the Drone RPP instances are huge, with up to 1680 vertices and 554 314 nonrequired edges, we are still capable of solving all except two of them in small computing times. For the two unsolved instances, the gap between the lower and upper bounds is very small (0.065% and 0.66%). These results confirm the comments made in Section 3.1 about the solution of the Drone RPP.

In most of these instances, RPP(0), RPP(1), RPP(3), and Drone RPP optimal solutions are the same, although there is a small improvement in some cases. Note that the values shown in this table correspond to the costs of the required and nonrequired edges traversed. While the nonrequired edges traversed may change for the solutions corresponding to the different RPP(k) and Drone RPP instances, the required ones are always traversed by any solution, and their cost represents a large part of the total cost. Therefore, this improvement in the cost would be more significant if we considered only the deadheading cost.

The reason for the optimal solutions to be so similar is that these random instances have many vertices incident with an odd number of required edges (about half of them). The deadheading in the optimal solutions contains a matching joining these odd-degree vertices and connects all the R-connected components. If there are many odd-degree vertices, the optimal solution uses nonrequired edges incident with these vertices to connect the components, not taking advantage of using the internal vertices of the lines. Therefore, the optimal solutions obtained for the Drone RPP with Assumptions 1 + 2—Drone RPP—are not very different from those obtained for the Drone RPP with Assumption 1—RPP(0)—only.

Table 5 shows the results obtained on the "Even" instances of Table 3. As expected, these instances are harder than the Random ones. In fact, two of the RPP(3) and 8 out of 15 Drone RPP instances could not be solved optimally in 1 hour of computing time, although RPP(0) and RPP(1) instances were still easily solved. We can observe that the differences in the solutions of the different RPP(k) and Drone RPP instances are more significant now, being especially noticeable in some cases like instance DroneRPP56. Overall, the behavior of the algorithm is very good, because, in spite of not being able to solve half of the instances to optimality, it has found feasible solutions for these instances that have been proved to be less than 0.6% from the optimal solution on average. The optimal solution of instance DroneRPP68 is depicted in Figure 7.

In many real utility networks (such as gas and electricity), at least at large scale, there can be many tree-like structures and odd degree nodes, as when lines terminate or when branch lines split off from a main line. These types of topologies (especially the odd degree nodes) create more opportunities for drones to fly off the network, thus providing greater benefits vs a vehicle

TABLE 4 Computational results on the "random" instances

	RPP(0)		RPP(1)		RPP(3)		Drone RPP	
Instance	**Opt.**	**Time**	**Imp. (%)**	**Time**	**Imp. (%)**	**Time**	**Imp. (%)**	**Time**
DroneRPP561	4347.09	0.11	0.00	0.15	0.00	0.30	0.00	8.17
DroneRPP562	5123.62	0.14	0.00	0.14	0.00	0.22	0.00	3.26
DroneRPP581	5929.69	0.13	0.00	0.14	0.00	0.37	0.00	9.62
DroneRPP582	7274.21	0.11	0.00	0.25	0.00	0.94	0.00	14.75
DroneRPP5101	6595.12	0.11	0.02	0.17	0.02	0.35	0.03	36.53
DroneRPP5102	10 259.65	0.39	0.00	0.48	0.00	0.83	0.00	19.98
DroneRPP661	4684.62	0.09	0.00	0.14	0.00	0.34	0.00	4.06
DroneRPP662	5298.72	0.08	0.00	0.16	0.00	0.30	0.00	13.44
DroneRPP681	6391.93	0.13	0.00	0.17	0.00	1.00	0.00	117.08
DroneRPP682	8797.12	0.19	0.00	0.35	0.00	0.81	0.00	20.50
DroneRPP6101	9263.24	0.17	0.03	0.27	0.03	0.53	0.05	23.68
DroneRPP6102	12 271.34	0.25	0.00	0.59	0.00	1.81	0.00	163.56
DroneRPP771	6654.00	0.22	0.00	0.23	0.00	1.25	0.00	744.85
DroneRPP772	10 441.47	0.41	0.00	0.27	0.00	0.94	0.00	5.75
DroneRPP791	10 039.24	0.22	0.00	0.48	0.00	1.78	0.07[a]	3608.68
DroneRPP792	11 994.28	0.28	0.39	0.55	0.39	0.97	0.40	55.77
DroneRPP7101	11 037.25	0.20	0.13	0.31	0.16	1.78	0.16	177.04
DroneRPP7102	13 440.90	0.30	0.12	0.38	0.12	0.72	0.12	27.05
DroneRPP881	10 294.90	0.19	0.30	0.23	0.30	0.88	0.30	41.93
DroneRPP882	12 538.79	0.22	0.00	0.80	0.00	11.30	0.00	217.18
DroneRPP891	10 990.88	0.50	0.72	0.64	0.72	2.41	0.72	3266.83
DroneRPP892	12 519.74	0.54	0.00	0.31	0.00	0.53	0.00	28.43
DroneRPP8101	12 496.34	0.36	0.00	0.83	0.00	5.36	0.65[a]	3616.09
DroneRPP8102	16 020.82	0.63	0.00	0.75	0.00	1.95	0.00	49.41
DroneRPP991	12 118.64	0.16	0.00	0.49	0.00	0.96	0.00	528.07
DroneRPP992	16 365.54	0.61	0.00	2.77	0.00	2.66	0.00	159.89
DroneRPP9101	13 283.21	0.27	0.00	0.43	0.00	3.53	0.00	46.13
DroneRPP9102	16 650.87	0.36	0.02	1.03	0.08	4.20	0.08	830.98
DroneRPP10101	15 001.44	0.22	0.00	0.86	0.03	1.28	0.03	129.41
DroneRPP10102	17 554.06	0.83	0.12	0.69	0.12	13.13	0.12	1284.55

[a]Means an upper bound.

FIGURE 8 A part of Spain's gas pipeline network [Colour figure can be viewed at wileyonlinelibrary.com]

that is restricted to the network. Figure 8 shows a map of part of Spain's gas pipeline network (from https://www.euractiv.com/section/energy/news/spanish-midcat-pipeline-to-replace-10-of-russian-gas-imports/), and the graph generated from this map, considering the area around Bilbao between Oviedo, Pamplona, and Segovia.

We created the Drone RPP instance corresponding to this network, which consists of a large component of 42 edges and 17 terminal nodes, and a second small component of 3 edges. We solved this (see Figure 9) and the obtained solution includes the drone flying 6 nonrequired edges within the large component, two nonrequired edges to connect the two components, and deadheading on 8 required edges. The solution corresponding to a vehicle restricted only to the network would consist of duplicating most of the edges in the instance, which is 41% longer than the Drone RPP solution.

TABLE 5 Computational results on the "even" instances

	RPP(0)		RPP(1)		RPP(3)		Drone RPP	
Instance	**Opt.**	**Time**	**Imp. (%)**	**Time**	**Imp. (%)**	**Time**	**Imp. (%)**	**Time**
DroneRPP56	5620.92	0.11	2.51	0.13	3.36	0.17	3.37	4.54
DroneRPP58	6637.89	0.16	0.48	0.29	0.49	1.02	0.53	730.97
DroneRPP510	9639.34	0.23	0.17	0.39	0.29	1.05	0.32	73.30
DroneRPP66	4871.30	0.13	0.01	0.31	0.17	1.25	0.21	661.78
DroneRPP68	7025.04	0.56	1.49	0.94	1.63	3.55	1.78	2164.14
DroneRPP610	9544.95	0.28	0.00	1.11	0.06	9.19	1.31[a]	3605.80
DroneRPP77	9700.92	0.36	0.32	1.64	0.35	1.92	0.45[a]	3610.62
DroneRPP79	12 088.01	0.35	0.07	0.47	0.07	2.14	0.09	1907.81
DroneRPP710	11 516.05	0.28	0.76	0.75	0.81	2.86	0.87	929.52
DroneRPP88	10 162.03	0.20	0.83	0.61	0.90	2.55	1.36[a]	3608.01
DroneRPP89	11 478.29	12.42	1.00	24.10	1.43[a]	3600.77	2.10[a]	3613.41
DroneRPP810	13 842.57	0.24	0.40	0.86	0.41	4.91	1.20[a]	3624.13
DroneRPP99	13 081.59	0.67	1.22	0.83	1.22	7.46	1.75[a]	3619.61
DroneRPP910	17 491.66	20.05	0.54	87.82	0.61[a]	3601.27	1.91[a]	3647.18
DroneRPP1010	18 336.60	0.58	0.05	5.78	0.13	336.21	0.97[a]	3680.60

[a]Means an upper bound.

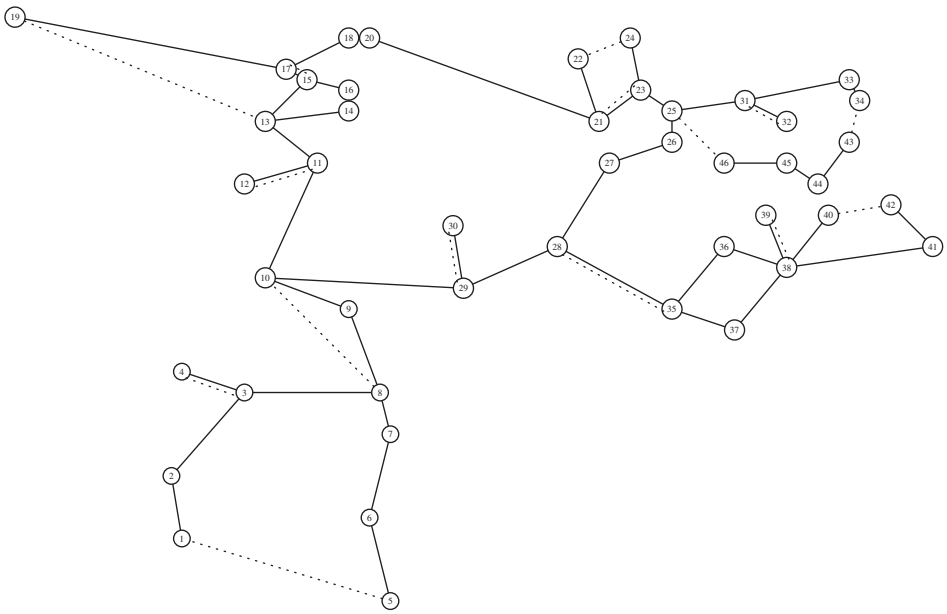

FIGURE 9 Solution of pipeline

5 | DRONES AND ARPS WITH SEVERAL VEHICLES

In this section, we make some comments on the case in which the limited capacity of the drones implies one vehicle alone cannot service all the required edges. Then, several drones may be needed. Let L be the time limit or maximum length of a single route, measured in terms of the edge costs. In this case, in addition to Assumptions 1 and 2, we can also assume that:

Assumption 3. The service of an edge can be shared by several vehicles.

In other words, a drone can enter an edge at a point, not necessarily an extreme point, travel and service a section of that edge, and leave it at another point. Other sections of this edge may be serviced by other drones. This assumption may be mandatory to obtain feasible solutions, as in the example in Figure 10.

Although in some instances it may not be mandatory, this assumption may be convenient to obtain better feasible solutions. Consider, for example, the instance in Figure 6A and suppose that the capacity of the drone is $L = 4000$. If we solve this instance

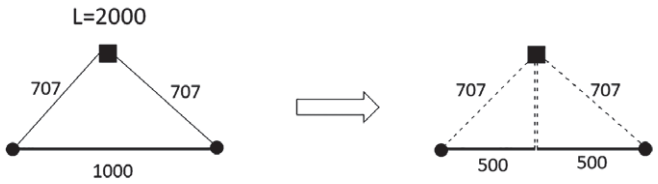

FIGURE 10 An instance where it is necessary to share the service of the edges

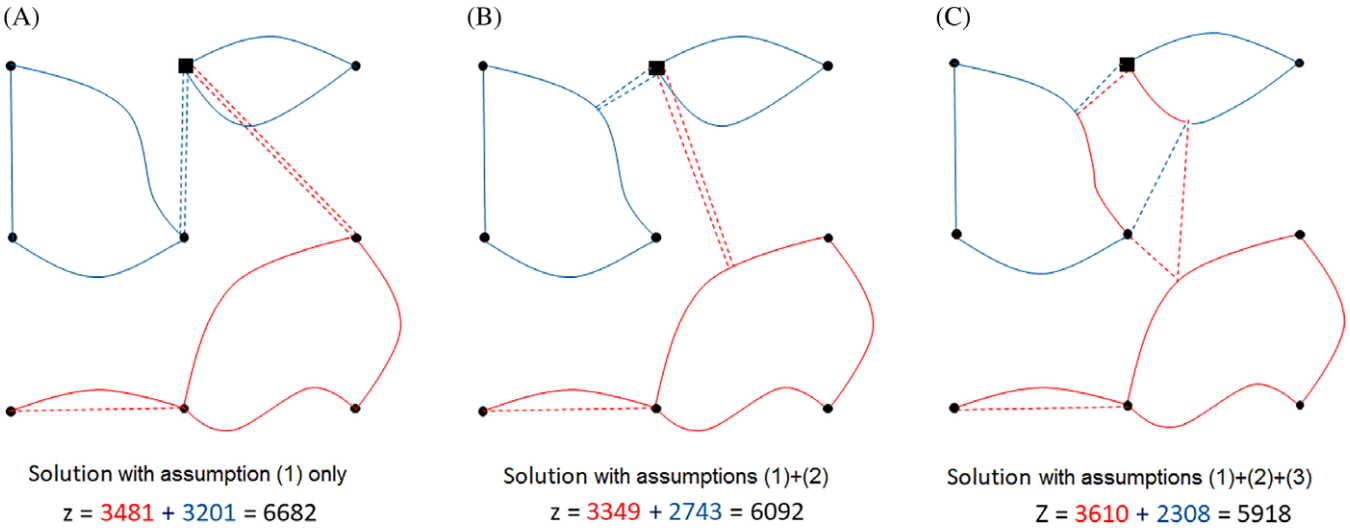

FIGURE 11 An instance where it is convenient to share the service of the edges [Colour figure can be viewed at wileyonlinelibrary.com]

with Assumption 1 only, we obtain the solution drawn in Figure 11A, consisting of two tours with lengths 3481 and 3201, respectively. The solution obtained with Assumptions 1 and 2, with tour lengths 3349 and 2743, is shown in Figure 11B. Finally, if we solve the instance with Assumptions 1 to 3, we obtain the solution drawn in Figure 11C, consisting of two tours with lengths 3610 and 2308. Note that these tours share the service of two edges, that is, there are two edges serviced by the two drones, each one traversing a different section of the edge.

With the Assumptions 1 to 3, we can define the problem for several vehicles as follows:

> **Definition 4.** (Length constrained k-drones rural postman problem, LCkDRPP) Given a set of lines, each one with an associated service cost, and a point called the depot, assuming that the cost of deadheading between any two points is the Euclidean distance, and given a constant L, find a set of tours starting and ending at the depot and with lengths no greater than L such that they jointly traverse all the given lines completely with minimum total cost.

In the LCkDRPP, the service of a given edge can be shared by two or more different vehicles. This feature also appears in other known routing problems such as the Split Delivery Vehicle Routing Problem (SDVRP). In the SDVRP, there is a set of customers, represented by vertices, each one with a given demand. These customers must be serviced by a fleet of vehicles with limited capacity. The demand of a customer can be split and serviced by several different vehicles. It is known that, if the costs satisfy the triangle inequality, there is an optimal solution to the SDVRP where no two routes have more than one customer in common [11]. However, in the LCkDRPP two vehicles can share more than one customer (edge), as shown in the previous example. Another simpler example is shown in Figure 12. Figure 12A shows the LCkDRPP instance with four required edges with cost 1000 and $L = 3000$, while Figure 12B depicts an optimal solution where two vehicles share two edges.

Another interesting point is that the number of drones needed to service an instance can be unbounded. Consider, for example, the instance shown in Figure 13. The depicted line is at a distance R from the depot and the maximum length of any route is $L = 2R + \varepsilon$. It is easy to see that in this case the number of drones needed to cover the line tends to infinity when ε tends to 0.

The LCkDRPP is much more difficult than the single vehicle version for several reasons. In addition to having several vehicles, the length constraints are difficult to handle. Furthermore, the number of nonrequired edges cannot be reduced as much as in the Drone RPP, because some of the rules described by Garfinkel and Webb [16] for the XPP and the RPP cannot be applied here. For example, the nonrequired edges joining two vertices, not both with odd degree, of the same R-connected component cannot be removed. The LCkDRPP is an area of ongoing research.

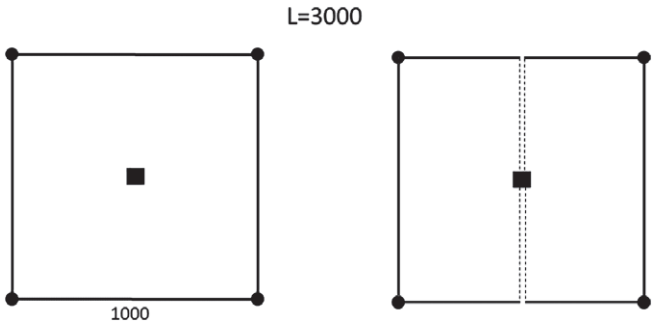

FIGURE 12 An instance where two drones share more than one customer (edge)

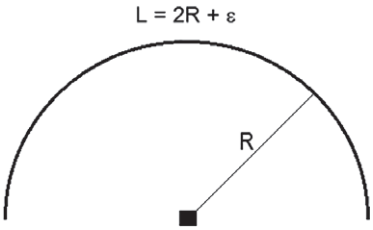

FIGURE 13 An instance where the number of drones needed is unbounded

6 | CONCLUSIONS

In this article, we have studied the Drone RPP and its relation with well-known postman ARPs. Also the relationship with cutting path problems has been discussed. Unlike postmen, whether they are walking or traveling with terrestrial vehicles, drones can travel directly between any two points in the plane (or in 3Ds) without following the edges of the network. This causes the Drone ARPs to possibly have better optimal solutions than postman ARPs. Particularly, we have shown that the worst-case ratio of the costs of postman ARPs with respect to that of Drone ARPs is infinite. However, Drone ARPs are continuous optimization problems with an infinite number of feasible solutions. In order to solve them as discrete optimization problems, we have approximated each line in the plane by a polygonal chain, thus allowing the vehicle to enter and leave each line only at the points of the polygonal chain. The resulting problem is a RPP. The higher the number of points in the polygonal chain, the better defined the line, but the size of the instance increases dramatically. We have proposed an algorithm that iteratively solves RPP instances with an increasing number of points.

We have generated a set of random Drone RPP instances, as well as some other instances that resemble real-life patterns. We have solved instances with up to 84 lines, whose corresponding RPP instances have up to 1680 vertices and more than 1 million edges.

Finally, we have also introduced the LCkDRPP, in which the limited capacity of the drones makes it impossible to service all the lines with one single vehicle. Therefore, we want to find a set of routes, each of limited length. The fact that drones can enter or leave the lines at any point makes it possible to have optimal solutions in which the service of a line is shared by several vehicles.

As future lines of research, we can consider several extensions. One extension would be to explore the 3D Drone RPP for inspecting infrastructure, such as complex bridges or an underwater oil platform. Note that [8] comments on the problem of periodical inspection of complex 3D structures, such as bridges or skyscrapers, that require checking the integrity of the surface of the beams in their skeleton. This problem can be modeled as a 2D instance of the windy RPP, although, to our knowledge, no real application of this kind has been solved using this approach. Other extensions include considering refueling/recharge points for the drones different from the depot, taking into account flight dynamics and their impact on energy or fuel consumption, the combination of trucks and drones (where trucks follow the network and are used to launch and recover drones on the network), and deciding the location of the depot for the drones combined with arc routing.

ACKNOWLEDGEMENTS

The work by Ángel Corberán, Isaac Plana, and José M. Sanchis was supported by the Spanish Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional (FEDER) through project MTM2015-68097-P (MINECO/FEDER).

ORCID

James F. Campbell https://orcid.org/0000-0003-2951-8703
Ángel Corberán https://orcid.org/0000-0002-0664-4232
Isaac Plana https://orcid.org/0000-0002-0516-4608
José M. Sanchis https://orcid.org/0000-0002-0039-8122

REFERENCES

[1] N. Agatz, P. Bouman, and M. Schmidt, *Optimization approaches for the traveling salesman problem with drone*, Transport. Sci. **52**(4) (2018), 739–1034.
[2] J. Boutilier, S. Brooks, A. Janmohamed, A. Byers, and J. Buick, *Optimizing a drone network to deliver automated external defibrillators*, Circulation **135** (2017), 2454–2465.
[3] T.C. Chan, H. Li, G. Lebovic, S.K. Tang, J.Y. Chan, H.C. Cheng, L.J. Morrison, and S.C. Brooks, *Identifying locations for public access defibrillators using mathematical optimization*, Circulation **127** (2013), 1801–1809.
[4] J.Y.J. Chow, *Dynamic UAV-based traffic monitoring as a stochastic arc-inventory routing policy*, Int. J. Transport. Sci. Technol. **5** (2016), 167–185.
[5] N. Christofides, V. Campos, Á. Corberán, and E. Mota, *An algorithm for the rural postman problem*, Imperial College Report, London, 1981.
[6] Á. Corberán and G. Laporte (eds), *Arc Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, Philadelphia, 2014.
[7] Á. Corberán, I. Plana, and J.M. Sanchis, *A branch & cut algorithm for the windy general routing problem and special cases*, Networks **49** (2007), 245–257.
[8] Á. Corberán, I. Plana, and J.M. Sanchis, "*The rural postman problem on directed, mixed, and windy graphs*," *Arc Routing: Problems, Methods, and Applications*, Á. Corberán and G. Laporte (eds), MOS-SIAM Series on Optimization, Philadelphia, 2014, pp. 101–128.
[9] R. Dewil, P. Vansteenwegen and D. Cattrysse, *A review of cutting path algorithms for laser cutters*, Int. J. Adv. Manuf. Technol. **87** (2016), 1865–1884.
[10] M. Dille and S. Singh, "*Efficient aerial coverage search in road networks*," *AIAA Guidance, Navigation and Control (GNC) Conference, August*, AIAA, Reston, VA, 2013. https://doi.org/10.2514/6.2013-5094.
[11] M. Dror and P. Trudeau, *Split delivery routing*, Naval Res. Logist. **37** (1990), 383–402.
[12] M. Dror (ed.), *Arc Routing: Theory, Solutions and Applications*, Kluwer Academic, Norwell, MA, 2000.
[13] L. Dubins, *On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents*, Am. J. Math. **79** (1957), 497–516.
[14] K. Easton and J.A. Burdick, "*Coverage algorithm for multi-robot boundary inspection*," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain*, IEEE, New York, 2005.
[15] L. Euler, *Solutio problematis ad geometriam situs pertinentis*, Comment. Acad. Sci. Petropolit. **8** (1736), 128–140.
[16] R.S. Garfinkel and I.R. Webb, *On crossings, the crossing postman problem, and the rural postman problem*, Networks **34** (1999), 173–180.
[17] B.L. Golden and R.T. Wong, *Capacitated arc routing problems*, Networks **11** (1981), 305–315.
[18] M. Guan, *Graphic programming using odd and even points*, Chin. Math. **1** (1962), 273–277.
[19] L.A. Haidari, S. Brown, M. Ferguson, E. Bancroft, M. Spiker, A. Wilcox, R. Ambikapathi, V. Sampath, D. Connor, and B. Lee, *The economic and operational value of using drones to transport vaccines*, Vaccine **34** (2016), 4062–4067.
[20] L. Hampton, 2008. In U.S. Gulf, robots, drones take on dangerous offshore oil work. https://www.reuters.com/article/us-world-work-oildrones/in-u-s-gulf-robots-drones-take-on-dangerous-offshore-oil-work-idUSKBN1I4100 (Accessed July 7, 2018).
[21] N. Ismail, 2017. Autonomous drones in the oil and gas industry. https://www.information-age.com/autonomous-drones-oil-gas-industry-123465606/ (Accessed July 7, 2018).
[22] N. Ismail, 2017. How can drones power the offshore oil and gas operations? https://www.information-age.com/can-drones-power-offshore-oil-gas-operations-123466745/ (Accessed July 7, 2018).
[23] M. Li, L. Zhen, S. Wang, W. Lv, and X. Qu, *Unmanned aerial vehicle scheduling problem for traffic monitoring*, Comput. Ind. Eng. **122** (2018), 15–23. https://doi.org/10.1016/j.cie.2018.05.039.
[24] T. Metcalfe, 2018. 24 underwater drones ? The Boom in Robotics Beneath the Waves. https://www.livescience.com/61431-underwater-drones.html (Accessed July 7, 2018).
[25] M.C. Mourão and L.S. Pinto, *An updated annotated bibliography on arc routing problems*, Networks **70** (2017), 144–194.
[26] C. Murray and A. Chu, *The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery*, Transport. Res. C: Emerg. Technol. **54** (2015), 86–109.
[27] H. Oh, H.S. Shin, A. Tsourdos, B.A. White, and P. Silson, "*Coordinated road-network search for multiple UAVs using Dubins path*," *Advances in Aerospace Guidance, Navigation and Control*, F. Holzapfel and S. Theil (eds), Springer, Berlin, Heidelberg, 2011, pp. 55–66.
[28] H. Oh, S. Kim, A. Tsourdos, and B.A. White, *Coordinated road-network search route planning by a team of UAVs*, Int. J. Syst. Sci. **45**(5) (2014), 825–840.
[29] C.S. Orloff, *A fundamental problem in vehicle routing*, Networks **4** (1974), 35–64.
[30] A. Otto, N. Agatz, J. Campbell, B. Golden and E. Pesch, *Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey*, Networks **72** (2018), 411–458.
[31] S. Poikonen, X. Wang, and B. Golden, *The vehicle routing problem with drones: Extended models and connections*, Networks **70** (2017), 34–43.
[32] B. Rabta, C. Wankmüller, and G. Reiner, *A drone fleet model for last-mile distribution in disaster relief operations*, Int. J. Disaster Risk Reduc. **28** (2018), 107–112.
[33] A.M. Rodrigues and J. Soeiro, *Cutting path as a rural postman problem: Solutions by memetic algorithms*, Int. J. Combin. Optimiz. Probl. Inform. **3** (2012), 22–37.
[34] J. Runyon, 2017. How drones will transform wind turbine inspections. https://www.renewableenergyworld.com/articles/print/volume-20/issue-6/features/wind/how-drones-will-transform-wind-turbine-inspections.html, (Accessed July 3, 2018).
[35] A. Sipahioglu, G. Kirlik, O. Parlaktuna, and A. Yazici, *Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach*, Robot. Auton. Syst. **58** (2010), 529–538.

[36] C. Warner, 2017. Submerged offshore pipeline inspections made easy with an ROV. https://www.deeptrekker.com/submerged-offshore-pipeline-inspections/ (Accessed July 3, 2018).
[37] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, *A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints*, IEEE Trans. Cybern. **44** (2014), 305–314.

How to cite this article: Campbell JF, Corberán Á, Plana I, Sanchis JM. Drone arc routing problems. *Networks.* 2018;72:543–559. https://doi.org/10.1002/net.21858