Received: 10 May 2022 | Revised: 5 July 2023 | Accepted: 11 July 2023

DOI: 10.1002/net.22176

RESEARCH ARTICLE

WILEY

The multi-purpose K-drones general routing problem

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

[1]Supply Chain and Analytics Dept., University of Missouri-St. Louis, St. Louis, Missouri, USA
[2]Dept. d'Estadística i Investigació Operativa, Universitat de València, Valencia, Spain
[3]Dept. de Matemáticas para la Economía y la Empresa, Universitat de València, Valencia, Spain
[4]Dept. de Matemática Aplicada, Universidad Politécnica de Valencia, Valencia, Spain

Correspondence
Paula Segura, Dept. de Matemática Aplicada, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain.
Email: psegmar@upvnet.upv.es

Funding information
Fondo Social Europeo (FSE), Grant/Award Number: PGC2018-099428-B-I00; European Regional Development Fund, Grant/Award Number: PID2021-122344NB-I00; European Social Fund; Ministerio de Ciencia e Innovación; Ministerio de Ciencia, Innovación y Universidades.

Abstract

In this article, we present and solve the multi-purpose K-drones general routing problem (MP K-DGRP). In this optimization problem, a fleet of multi-purpose drones, aerial vehicles that can both make deliveries and conduct sensing activities (e.g., imaging), have to jointly visit a set of nodes to make deliveries and map one or more continuous areas. This problem is motivated by global healthcare applications that deploy multipurpose drones that combine delivery trips with collection of aerial imaging data for use in emergency preparedness and resilience planning. The continuous areas that have to be mapped may correspond to terrain surfaces (e.g., flooded areas or regions with a disease outbreak) or to infrastructure networks to be inspected. The continuous areas can be modeled as a set of lines so that each area is completely serviced if all the lines covering it are traversed. Thus, given a set of nodes and a set of lines, the problem is to design drone routes of shortest total duration traversing the lines and visiting the nodes, while not exceeding the range limit (flight time) and capacity (loading) of the drones. Unlike ground vehicles in classical routing problems, drones can enter a line through any of its points, service only a part of that line and then exit through another of its points. The possibility of flying directly between any two points of the network offered by drones can lead to reduced costs, but it increases the difficulty of the problem. To deal with this problem, the lines are discretized, allowing drones to enter and exit each line only at a finite set of points, thus obtaining an instance of the K-vehicles general routing problem (K-GRP). We present in this article an integer programming formulation for the K-GRP and propose a matheuristic algorithm and a branch-and-cut procedure for its solution. Results are provided for problems with up to 20 deliveries and up to 28 continuous areas.

KEYWORDS

branch and cut, capacity constraints, drones, general routing problem, logistics, matheuristic, max-time constraints

1 | INTRODUCTION

Drones or UAVs (unmanned aerial vehicles) are unmanned flying vehicles that can be flown autonomously or controlled remotely. Commercial drones are deployed for a wide variety of missions, with two broad categories being delivery and sensing. Delivery drones can be used to deliver items to discrete locations, such as for packages, medical items (including blood, organs, defibrillators, medicines, vaccines, etc.) and disaster relief supplies. Another type of drone delivery in agricultural applications involves delivery of liquid chemicals (e.g., herbicides, fertilizers) or seeds and seedlings to a field or region [10]. Note that this type of agricultural delivery operation has more in common with agricultural drone sensing operations, as in both a continuous region of earth is involved (e.g., a farm field) rather than the discrete delivery points for package delivery. Drones for sensing operations such as mapping, surveillance, inspection, and search activities are equipped with environmental sensors (such as optical cameras, multispectral cameras, thermal sensors, LIDAR, radars of various types, magnetometers, etc.) to collect data about ground, underground or infrastructure conditions (e.g., plant health, buried objects, wild fires, flooding, road traffic, bridge conditions, etc.). Drones are also used in sensing operations to collect data on the air and weather conditions through which they fly, such as pollution, temperature, humidity, particulates, volatile organic compounds and so forth. Drone sensing operations are accomplished by flying the drone over a region of the earth or an item of infrastructure in a path that provides coverage of the area of interest. The region to be covered may be a 2- or 3-dimensional surface (e.g., a bounded terrain area or a bridge) or a network defined by infrastructure (e.g., roads, power transmission lines, or pipelines). Drones have several benefits for sensing and delivery missions compared to "manned" operations, including cost reduction and time savings, ability to access difficult-to-reach areas, and ability to operate in settings that are too dangerous for a person, or that require a level of accuracy that only technology achieves.

Drone flights have typically been designed for a single purpose (e.g., either mapping or delivery), though some drones can be reconfigured to accomplish different purposes on different trips (e.g., [1, 14]). Thus, it is important to distinguish between a drone that can be (re)configured to perform different tasks (delivery, mapping, etc.) on separate trips, and a drone that can perform multiple tasks (e.g., both delivery and mapping) on the same trip. In this article "multi-purpose drones" refers to the latter case. In a typical drone delivery, the drone makes an empty return trip after the last delivery (assuming all deliveries are successful) and this deadheading seems inherently wasteful. If such a drone could be used for other activities such as mapping or sensing, then the return trips would be less wasteful. Similarly, if a drone being used for sensing was able to make a delivery near the flight path needed for sensing, then efficiencies might be gained. Optimizing the routing of drones for flights that combine delivery and sensing in one trip is the key issue addressed in our research.

As drone capabilities expand, multipurpose drone trips have become a possibility. Our research was originally motivated by a UNICEF project in Malawi for multi-purpose drones that could both make deliveries of healthcare items, and perform imaging as well ([32, 33]). The imaging may be for mapping a flooded area, an area with a disease outbreak or any other type of continuous area, while the delivery would be to one or more discrete nodes. The continuous area to be imaged (mapped or photographed) is covered by a series of drone flights so that images are obtained that cover the entire region of interest. In many settings, repeated imaging of a region is required to assess the changing conditions, whether the application area is for plant health, fire spread or flooding.

There are some reported applications of multipurpose drone trips in global healthcare settings. Drone provider Swoop Aero, in association with the UK Department for International Development and UNICEF Malawi, studied the benefits of using multi-purpose drones in Malawi to improve access to healthcare for remote communities and improve disaster preparedness through aerial mapping [39]. In 2020, they undertook a 10-month sustained multi-purpose air medical logistics and disaster relief operation in two districts of Malawi to combine daily long-range drone deliveries with flood mapping and disaster response. A report on this project [29] says that "Swoop Aero successfully proved the multifunctionality of the technology-based platform to conduct simultaneous aerial mapping tasks as well as routine medical commodity deliveries within the south of the country". Another example of multipurpose drone trips in global health is [40] which notes that a drone delivering medical supplies could also be equipped with a sensor to obtain agricultural data from farms the drone passes on the way, serving two different users at once. Another project in Malawi is described in [41] in which a single UAV performs three separate public services: medical supply, soil mapping for UNICEF (inspection and imaging that could be used for agriculture, infrastructure and development projects), and aerial surveillance for the Malawi Ranger Service to monitor endangered species and detect poaching. A very different example of multipurpose drone trips that includes sensing and collection (instead of delivery) is [12] which describes using a drone for sensing near erupting volcanoes. Here the drone has sensors to measure concentrations of various molecules and environmental conditions in the volcanic plume, and it also collects physical samples of the volcanic plume.

There are also applications for multipurpose drones that focus on communication of the data acquired in the sensing operations. In these cases, the delivery component is downloading (or uploading) information, which is accomplished by having the drone travel to close proximity to a base station or other drone. Reference [17] considers a system with one drone route that conducts video surveillance over a rectangular region using repeated parallel passes of the drone, followed by travel to a delivery location to communicate the information collected to a base station. This research focuses on specifics of the communication channels and communication protocols, rather than the drone routing. Reference [9] considers a similar problem with surveillance using a camera of a "long strip" as for linear infrastructure (e.g., power transmission line, roadway or pipeline), followed by delivery of the captured images to a base station. This includes problems where a single drone can surveille the width of the strip in one pass, and an extension where multiple drones fly together down the length of the strip (as each drone's camera can cover only part of the width of the strip). Routing for the sensing component is simplified by following one linear feature, and the research focuses on the data transmission to the base station node with data transmission rate constraints. Combining sensing and delivery of information (communication) to a base station is a growing area of research as “there is huge interest to perform joint communications and sensing now” [21].

Multipurpose drones trips are also used in some other settings including avalanche scenarios, drowning, and search and rescue. Reference [19] describes using drones for sensing to detect an area likely to be avalanched over a road, and then having the drone accurately drop explosive charges (delivery) to trigger controlled avalanches. In [28] a drone is used to thermally detect a body buried in an avalanche and then to drop an avalanche beacon nearby. A similar use of drones is reported in [6] for search and rescue in the Canadian Arctic, where the drone can drop a communication device to a lost party once located. Finally, drones for rescue of swimmers in distress is reported in [27], where there is testing of drones that first search for drowning swimmers and then deliver a flotation device. All of these applications involve sensing to find something or someone, and then delivery (dropping) of an item. These are different than the global health and communications applications described above in that the delivery is made to a location within the search area, and the delivery location is not known at the outset of the drone flight.

There is very limited analytical research on routing in multi-purpose drone problems that combine delivery with sensing activities in the same drone trip. In [16], the authors study a problem in which the aim is to design drone trajectories that perform a transportation operation such as package delivery, while also providing uniform coverage (over time) of a neighborhood area. They investigate the use of multi-purpose drones in a simplified scenario where the neighborhood area is a circular region, and in another scenario where the area is an arbitrarily shaped region. The authors propose an algorithm for each scenario for uniform coverage and last-mile delivery applications.

In this article we address the multipurpose drone general routing problem where K drone routes are to be determined to both provide sensing over a defined region and make a set of deliveries to discrete points. The region needing sensing coverage may be a two-dimensional surface (e.g., a flooded region) or a network of linear features (e.g., rods or power transmission lines). Sensing by a single drone covers a linear swath as the drone flies with the width of the swath (and the resolution) determined by the sensor and the altitude of the drone. In our research linear features to be covered by sensing are modeled as line segments which are covered by a single drone traversal along the line segment. To cover two dimensional regions by the drone, we replace the region by a set of parallel lines to be flown by the drone that provide complete coverage over the region of interest (i.e., the swaths from the drone flights cover the region). Typically the parallel lines are oriented to minimize the number of turns by the drone, as turns interrupt the data collection and increase the drone flight distance and time (and battery usage) [30]. Because we allow a region to be covered on several different drone trips, the optimal orientation of the flights might be different for different parts of the region. We adopt the reasonable strategy for covering a region to have the drones fly parallel lines oriented with the longer axis of the region (thus minimizing the turns needed by the drones), where the spacing of the lines reflects the characteristics of the sensor being used by the drones. Although finding the optimal way of designing these parallel lines is an interesting and difficult problem itself, in this work we will assume that the line segments that the drones have to follow in order to provide sensing over the areas are parallel, straight and already given. However, our drone routing methodology will work for any set of given lines.

Thus, given a set of lines, whether a network or lines covering a two-dimensional region, and a set of delivery points with a known demand, the problem addressed here consists of designing drone routes of lowest total duration, jointly traversing all the lines and visiting all the points (to make deliveries). The duration of each route should not exceed a time limit and the demand delivered by each route should not exceed the capacity of the drone.

Other articles have studied the routing of drones to provide sensing (i.e., cover) over one or several areas, but without including any drone deliveries. Xie et al. [37] study a path planning problem consisting of finding the optimal tour for a single drone that has to cover multiple separated convex polygonal regions. The authors provide a mixed integer programming formulation and propose a procedure for covering a single convex polygonal region. Based on this method, they develop two approaches to solve the complete problem, a dynamic programming-based exact algorithm and a heuristic to generate high-quality tours. Puerto and Valverde [26] address the problem of designing routes for drones that must visit a number of geographical elements to deliver some good or service. They present two formulations that are tested on a set of instances with different shapes of elements, second order cone (SOC) representable and polyhedral neighborhoods and polygonal chains. Yang et al. [38] studied the construction of a route for a drone that has to map an area while minimizing the total distance. The authors split the area into squares and present an ant colony metaheuristic to design the route that visits them. Wang et al. [36] consider the routing of drones that must “supervise” disjoint areas over a given time horizon. The areas are divided into cells which must be visited several times within a time period. A multiobjective evolutionary algorithm to solve the problem is proposed. Vasquez-Gomez et al. [34] propose a method for finding a path for a drone covering several disjoint areas consisting of two steps: first determining the visiting order of the areas and then the optimization of the flight lines orientation. In Vasquez-Gomez et al. [35], the authors address the same problem with only one region but taking into account the starting and ending points of the drone.

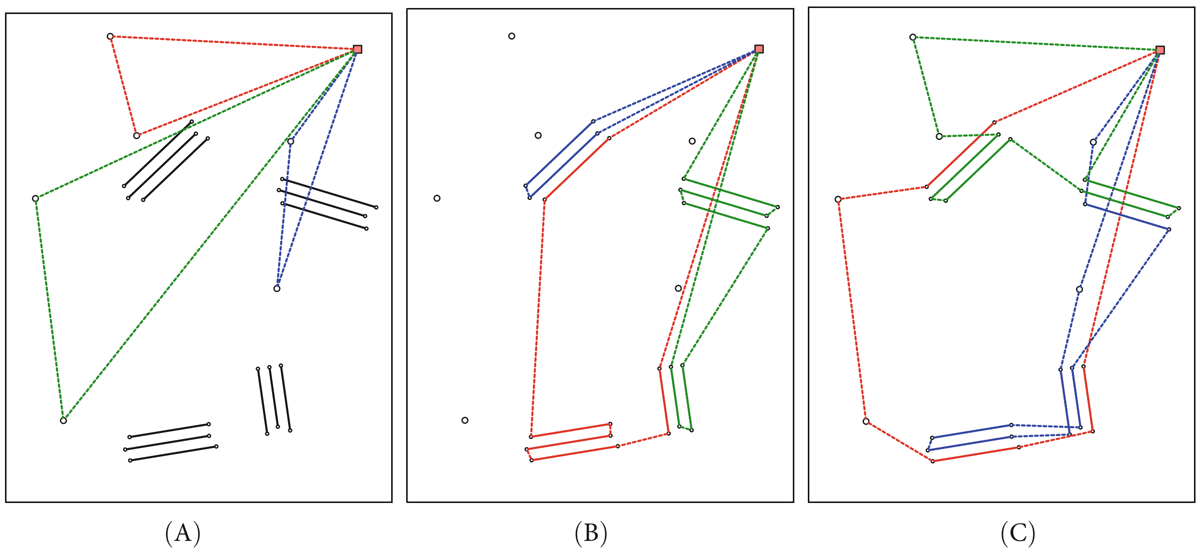


FIGURE 1 Single-purpose drones versus multi-purpose drones. (A) Delivery routes, (B) imaging routes, (C) multi-purpose routes.

In this work, we address a routing problem for multi-purpose drones that both make deliveries and also do some sensing on the same drone trip. Thus, rather than have one trip for deliveries and a second trip for sensing (using a reconfigured or different drone), a single drone trip might perform both deliveries and sensing. Using multi-purpose drones instead of single-purpose ones can lead to considerable improvements in the total duration of the trips, as can be seen in the example illustrated in Figure 1. This example includes six delivery points and four regions represented by four sets of parallel lines that provide coverage of four regions. Figure 1A shows the optimal routes for three delivery-only drone trips delivering packages to the six customers, with a total duration of 3234.65. Figure 1B presents the optimal routes of three sensing-drone trips covering the four areas, whose total duration is 4361.63. However, using three multi-purpose drone trips, as can be seen in Figure 1C, the delivery and sensing can be carried out with a total duration of 5574.07, which is 26.6% lower than for the six drone trips using single-purpose drones.

The contributions of this research are: (1) introducing and solving a new drone routing problem for multipurpose drone trips motivated by several applications, including in global healthcare and sensing/surveillance; (ii) development of a general solution methodology for multiple drones that allows an area to be covered by several different drone trips, (iii) computational results to document performance of the method and benefits over use of two kinds of single purpose drones. Note that our approach allows sensing coverage of two-dimensional areas covered by any linear features (whether parallel line segments or not), as well as coverage of networks and linear features themselves, as for infrastructure (transmission lines, pipeline, roadways, etc.).

In Section 2, we formally describe the multi-purpose K-drones general routing problem, propose a formulation for its discrete version, the K-vehicles general routing problem (K-GRP), and present several valid inequalities for this problem. In Section 3, a branch-and-cut algorithm for the solution of the K-GRP is proposed, while Section 4 is devoted to the description of a matheuristic for the MP K-DGRP. The computational experiments are presented in Section 5 and some conclusions and future research lines are discussed in Section 6.

2 | THE MULTI-PURPOSE K-DRONES GENERAL ROUTING PROBLEM

Let us consider a finite family $\mathcal{A}$ of separated continuous areas, a set $\mathcal{V}$ of isolated locations and a fleet of multipurpose drones located at a depot v_0 (see left image in Figure 2). Multipurpose drones have both imaging and delivery capability, and they have to provide sensing over the continuous regions in $\mathcal{A}$ and also make a certain delivery at each location in $\mathcal{V}$, without running out of battery and without exceeding their load capacity. We assume that each region or area $A \in \mathcal{A}$ is described (covered) by a set of virtual parallel straight lines so that, by traversing these lines, drones cover the whole area (see right image in Figure 2). The separation distance between the parallel lines depends on technical aspects of the drone and its sensing equipment (such as the size of the camera footprint, the altitude of the flight of the drones, etc.). Moreover, drones are capable of delivering goods to the locations of $\mathcal{V}$, assuming that the load they can transport is limited and the demand of each location must be delivered by the same vehicle at one time.

The MP K-DGRP can be defined as follows. Let us consider a set of lines, each one with an associated service (traversing) time, a depot v_0, and a set of locations with an associated positive demand and a service (visiting) time, and assume that the

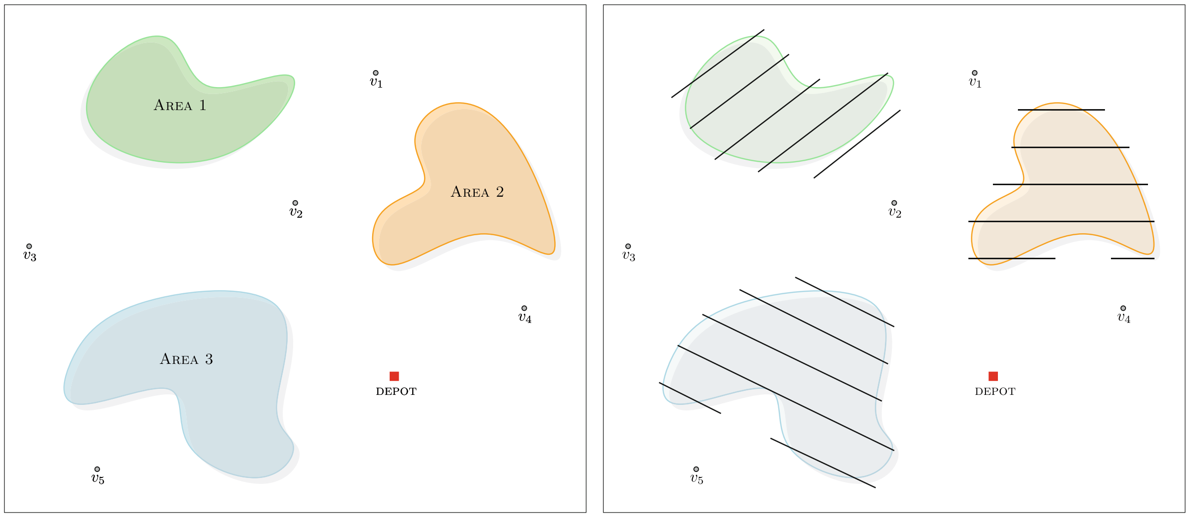


FIGURE 2 A multi-purpose drone framework with $|\mathcal{A}| = 3$ and $|\mathcal{V}| = 5$.

time of deadheading between any two points is the Euclidean distance. Given two constants L and Q, the problem consists of finding a set of drone routes starting and ending at the depot, with duration no greater than L and load not exceeding Q, such that they jointly traverse all the given lines and satisfy the demand of all the locations with minimum total duration. In this problem, we consider that a drone can travel in a straight line between any two points, and not necessarily following the links of the given network as happens in classical routing problems with ground vehicles. Thus, a drone can enter a line that requires service through any of its points, traverse and service part of it, exit the line through another of its points, then travel directly to another line or to make a delivery at a required location, and continue its route while the time limit L is not exceeded.

The ability of drones to enter or leave each line at any of its points allows better solutions (lower cost routes) than those obtained with traditional vehicles that cannot travel off of a network, but it makes the optimization problem continuous and very difficult to address. To deal with this issue, as is done in [4] and [5], we select for each instance a finite number of points on each line, so that drones can enter and leave each line only at these points. In other words, each original line is divided into small edges that must be traversed by the drones. In this way, for example, a single edge with two endpoints may have four added intermediate points that transform the one original edge into five edges, each with time of traversal equal to the proportional part of the time of traversal of the corresponding original edge. Thus, the continuous problem is reduced to a discrete problem, the K-vehicles General Routing Problem (K-GRP).

Obviously, the more intermediate points selected on each line, the better is the approximation of the original continuous problem, and thus better solutions might be obtained. However, if the number of intermediate points is very large, the size of the instance increases so much that it cannot be addressed, and even heuristic algorithms can fail to provide good solutions in reasonable computing time. Thus, it is necessary to devise sophisticated strategies to generate instances of the K-GRP with a reduced but significant number of intermediate points that can be solved to produce good solutions for the MP K-DGRP.

In what follows, we formally define the K-GRP and propose a formulation for it.

2.1 | A formulation for the K-vehicles general routing problem

Given an undirected and connected graph $G = (V, E)$, a subset $E_R \subseteq E$ of required edges and a subset $V_R \subseteq V$ of required vertices, the General Routing Problem (GRP, Orloff [22]) consists of finding a minimum cost tour (closed walk) on G traversing each edge $e \in E_R$ and visiting each vertex $v \in V_R$ at least once.

In the K-vehicles General Routing Problem (K-GRP), the required edges $E_R \subseteq E$ correspond to the segments into which we have split the straight lines that have to be traversed in order to completely cover the continuous areas. Since these lines are isolated straight lines, that is, there are no lines adjacent to each other, when they are discretized we obtain isolated chains of required edges. Therefore, the vertices incident with edges in E_R are incident with either one required edge (we call this set of vertices V_O) or with two required edges (we call this set of vertices V_E). We denote by E_{NR} the set of nonrequired edges, which form a complete graph with the node set V. There is a time $c_e^s \geq 0$ associated with the traversal and service of each required edge $e \in E_R$, and a deadheading time $c_e \geq 0$ associated with the traversal of each nonrequired edge $e \in E_{NR}$. The deadheading times satisfy the triangular inequality. The set of required vertices $V_R \subseteq V$ is formed by all the nodes in V where a drone should make a delivery. We will assume that these vertices are only incident with nonrequired edges and have to be serviced by exactly one drone. Otherwise, a simple transformation can be applied to meet this condition. Each node $i \in V_R$ has an associated positive integer demand $d_i > 0$ and a service time $c_i \geq 0$. Furthermore, the drone routes start and end at a vertex (the depot, vertex v_0) that, for the sake of simplicity, we will assume is not incident with required edges. Hence, we have $V = \{v_0\} \cup V_R \cup V_O \cup V_E$.

There is a fleet of K drones, each with a load capacity Q and a time limit L. The objective of the K-GRP is to find K routes with minimum total duration, starting and ending at the depot, that jointly traverse all the required edges and visit the required vertices exactly once, so that the duration and the load of each route does not exceed the values L and Q, respectively. We will call a *K-GRP solution* any set of K tours that meet all the constraints of the problem.

The K-GRP can be formulated using a binary variable x_e^k for each edge $e \in E_R$ and for each drone $k \in \{1, \dots K\}$, and two binary variables x_e^k and y_e^k for each edge $e \in E_{NR}$ and for each drone k. Variable x_e^k takes the value 1 if the required edge e is traversed (and serviced) by drone k and 0 otherwise, while variables x_e^k and y_e^k take the value 1 if the nonrequired edge e is traversed once or twice by drone k, respectively, and 0 otherwise. In other words, variables x_e^k and y_e^k represent the first and second traversal of nonrequired edge e by drone k. Additionally, a binary variable z_i^k is introduced for each vertex $i \in V_R$ and each drone k, taking the value 1 if the vertex i is serviced by drone k and 0 otherwise.

We use the following notation. Given two subsets of vertices $S, S' \subseteq V$, $(S : S')$ denotes the edge set with one endpoint in S and the other in S'. Given a subset $S \subseteq V$, let us denote $\delta(S) = (S : V \setminus S)$ and let $E(S) = \{e = (i,j) \in E : i,j \in S\}$ be the set of edges with both endpoints in S. For any subset $F \subseteq E$, we denote $F_R = F \cap E_R$, $F_{NR} = F \cap E_{NR}$, $x^k(F) = \sum_{e \in F} x_e^k$ and $(x^k + y^k)(F) = \sum_{e=(i,j) \in F} (x_e^k + y_e^k)$.

We propose the following formulation for the K-GRP:

$$\text{Minimize } \sum_{k=1}^{K} \sum_{e \in E_{NR}} c_e \left(x_e^k + y_e^k\right) + \sum_{k=1}^{K} \sum_{e \in E_R} c_e^s x_e^k + \sum_{k=1}^{K} \sum_{i \in V_R} c_i z_i^k, \tag{1}$$

s.t.

$$\sum_{e \in \delta_R(i)} x_e^k + \sum_{e \in \delta_{NR}(i)} \left(x_e^k + y_e^k\right) \equiv 0 \pmod 2, \qquad \forall i \in V \setminus V_R, \quad \forall k, \tag{2}$$

$$\sum_{e \in \delta(i)} (x_e^k + y_e^k) = 2\, z_i^k, \qquad \forall i \in V_R, \quad \forall k, \tag{3}$$

$$\sum_{k=1}^{K} x_e^k = 1, \qquad \forall e \in E_R, \tag{4}$$

$$\sum_{k=1}^{K} z_i^k = 1, \forall i \in V_R, \tag{5}$$

$$x_e^k \geq y_e^k, \qquad \forall e \in E_{NR}, \quad \forall k, \tag{6}$$

$$\sum_{e \in \delta_R(S)} x_e^k + \sum_{e \in \delta_{NR}(S)} \left(x_e^k + y_e^k\right) \geq 2 x_f^k, \qquad \forall S \subseteq V \setminus \{v_0\}, \forall f \in E(S), \forall k, \tag{7}$$

$$\sum_{e \in E_R} c_e^s x_e^k + \sum_{e \in E_{NR}} c_e \left(x_e^k + y_e^k\right) + \sum_{i \in V_R} c_i z_i^k \leq L, \qquad \forall k, \tag{8}$$

$$\sum_{i \in V_R} d_i z_i^k \leq Q, \qquad \forall k, \tag{9}$$

$$z_i^k \in \{0,1\}, \qquad \forall i \in V_R, \quad \forall k, \tag{10}$$

$$x_e^k \in \{0,1\}, \qquad \forall e \in E_R, \quad \forall k, \tag{11}$$

$$x_e^k, y_e^k \in \{0,1\}, \qquad \forall e \in E_{NR}, \quad \forall k. \tag{12}$$

The objective function (1) minimizes the total duration of the routes. The first term represents the deadheading time from traveling on nonrequired edges. The second and third terms represent, respectively, the time of servicing the required edges and the time of visiting the required nodes (for delivery). Due to constraints (4) and (5), both these last two terms are constant and could be removed from the objective. Constraints (2) force that the number of times a drone visits a nonrequired vertex is even, possibly zero, and equalities (3) ensure that a drone traverses two nonrequired edges incident with a required vertex i if it is serviced by this drone (and traverses no incident edges if it is not serviced). Equations (4) force each required edge to be serviced exactly once, and the visit of each required vertex by exactly one drone is ensured by constraints (5). Constraints (6) guarantee that a second traversal of a nonrequired edge by a drone can only occur when it has been traversed previously by this drone. Inequalities (7) avoid subtours and ensure that each single route is connected and connected to the depot. Constraints (8) guarantee that the duration of each route does not exceed L, while constraints (9) ensure that the demand serviced on each route is not greater than the drone capacity Q. Constraints (10), (11) and (12) are the binary conditions for the variables.

The following results allow us to remove some variables from the formulation, taking into account the structure of the optimal K-GRP solutions.

Theorem 1. *There is an optimal K-GRP solution in which each route satisfies*:

(a) It does not deadhead two edges (i,j), (j,k) consecutively, except if $j \in V_R$ and it is serviced by the route.
(b) It does not visit the vertices in $\{v_0\} \cup V_R \cup V_O$ more than once.
(c) It does not visit the vertices in V_E more than twice.

Proof. (a) Let us suppose a tour deadheads two edges (i,j), (j,k) consecutively with $j \notin V_R$. Each one of these two edges is either a nonrequired one or a required edge that had already been serviced in a previous traversal. Recall that the nonrequired edge (i,k) exists because E_{NR} induces a complete graph, and that $c_{ik} \leq c_{ij} + c_{jk}$ due to the triangle inequality. The tour obtained after replacing the traversal of (i,j), (j,k) with the traversal of (i,k) has a lower or equal duration and services the same required edges and the same required vertices as the former tour. By iterating this argument we obtain (a).

(b) The drone tours start and end at v_0. If a tour visits v_0 again, it deadheads two edges without servicing v_0, which contradicts (a). Let us suppose a tour visits a given vertex $j \in V_R \cup V_O$ more than once. Note that there is only one service associated with vertex j: the service of j, if $j \in V_R$ or the service of the unique required edge incident with j, if $j \in V_O$. Therefore, all visits to the vertex j, except one of them, are performed by deadheading two edges without servicing j, which contradicts (a).

(c) Let us suppose a tour visits a given vertex $j \in V_E$ more than twice. Note that there are only two services associated with vertex j: the service of the two required edges incident with $j \in V_E$. Therefore, all visits to the vertex j, except two of them, are performed by deadheading two edges without servicing j, which contradicts (a).■

Corollary 1. *For each drone k, variables y_e^k for the following nonrequired edges e can be removed from the formulation*:

- *$e = (u,v)$ with $u \in V_O$ (or $v \in V_O$), and*
- *$e = (u,v)$ with $u, v \in V_R$.*

and the following inequalities can be added to the formulation:

$$x^k(\delta(i)) = 2x_{ij}^k, \quad \forall (i,j) \in E_R, \ \textit{with } i \in V_O, \forall k, \tag{13}$$

$$x^k(\delta(i)) \leq 2x_{ij}^k + 2x_{li}^k, \quad \forall (i,j),(l,i) \in E_R \ (i \in V_E), \ \forall k. \tag{14}$$

Note: Consider the special case of the K-GRP in which the lines are not split, that is, $V_E = \emptyset$. From Corollary 1, this case can be formulated using variables y_e^k only for those edges joining the depot and vertices in V_R (all the other y_e^k variables can be removed). Moreover, from Theorem 1, there is an optimal solution that does not use any nonrequired edge parallel to a required one (otherwise, two consecutive edges would be deadheaded), so no variable x_e^k is needed for these edges.

2.2 | Other valid inequalities for the K-GRP

In this section we present four families of valid inequalities that reinforce the formulation. They are based on the capacity and autonomy constraints of drones or on known families of valid inequalities proposed in [7] for the Maximum Benefit Chinese Postman Problem (MBCPP).

Parity inequalities are obtained from those proposed in [7] for the MBCPP, which generalize the well known co-circuit inequalities [2]. They rely on the fact that each drone tour traverses any edge cut-set an even (or zero) number of times:

$$x^k(\delta_R(S) \setminus F_R) + (x^k - y^k)(\delta_{NR}(S) \setminus F_{NR}) \geq x^k(F_R) + (x^k - y^k)(F_{NR}) - |F| + 1, \tag{15}$$

for each drone $k \in \{1, \dots K\}$, for all $S \subset V$, and for all $F \subseteq \delta(S)$ with $|F|$ odd.

In order to see that inequalities (15) are valid for the K-GRP on G, note that $x^k(F_R) + (x^k - y^k)(F_{NR}) \leq |F|$. For a drone k, the tours that satisfy $x^k(F_R) + (x^k - y^k)(F_{NR}) = |F|$ traverse each edge in F once, and, since $|F|$ is odd, they traverse at least one edge in $\delta(S) \setminus F$ once, thus satisfying $x^k(\delta_R(S) \setminus F_R) + (x^k - y^k)(\delta_{NR}(S) \setminus F_{NR}) \geq 1$. For the drone tours satisfying $x^k(F_R) + (x^k - y^k)(F_{NR}) \leq |F| - 1$, inequality (15) reduces to $x^k(\delta_R(S) \setminus F_R) + (x^k - y^k)(\delta_{NR}(S) \setminus F_{NR}) \geq 0$, which is obviously satisfied.

p-connectivity inequalities are based on those with the same name proposed for the MBCPP in [7] and are generalized here to also consider the existence of required vertices. Let $\{S_0, \dots, S_p\}$ be a partition of V. Assume that $v_0 \in S_d$, $d \in \{0, \dots, p\}$, and select either an edge $e_j \in E_R(S_j)$ or a vertex $v_j \in S_j \cap V_R$ for each $j \in \{0, \dots, p\} \setminus \{d\}$. Let I_1 and I_2 be the sets of indices in $\{0, \dots, p\} \setminus \{d\}$ in which a required edge or a required vertex has been selected, respectively. For each drone k, the following inequality is valid.

$$x^k(\delta_R(S_0)) + (x^k + y^k)(\delta_{NR}(S_0)) + 2\sum_{1\le r<t\le p} x^k(S_r : S_t) \ge 2\sum_{j\in I_1} x^k_{e_j} + 2\sum_{j\in I_2} z^k_{v_j}. \tag{16}$$

Let us assume that the depot belongs to S_0 and let $(\overline{x}, \overline{y}, \overline{z})$ be a K-GRP tour.

Given a drone k, if there is an edge $e_j, j \in I_1$, or a vertex $v_j, j \in I_2$, such that $\overline{x}^k_{e_j} = 0$ or $\overline{z}^k_{v_j} = 0$, we can consider another p-connectivity inequality with $p-1$ subsets by merging sets S_j and S_{j+1} (or S_{j-1}). If the new $(p-1)$-connectivity inequality is satisfied by the solution, the original one will also be. Therefore, we can assume that $\overline{x}^k_{e_i} = 1$ for all $i \in I_1$ and $\overline{z}^k_{V_O} = 1$ for all $i \in I_2$.

Moreover, if $\overline{x}^k(S_r : S_t) \ge 1$, we can also merge S_r and S_t, obtaining a new $(p-1)$-connectivity inequality. As before, if this inequalty is satisfied, so is the original one. Therefore we can also assume that $\overline{x}^k(S_r : S_t) = 0$ for any pair of sets S_r, S_t.

Since each set S_i, $i > 0$, must be connected to the depot and $\overline{x}^k(S_r : S_t) = 0$, $x^k(\delta_R(S_0)) + (x^k + y^k)(\delta_{NR}(S_0)) \ge 2(|I_1| + |I_2|)$, and the inequality holds.

Capacity inequalities are widely used in node and arc routing problems when there is a limit to the demand a vehicle can service. Given a vertex set $S \subseteq V \setminus \{v_0\}$, the edge cut-set $\delta(S)$ has to be traversed at least twice the number of vehicles needed to service the demand of the required vertices in S, and, therefore, the following inequality is valid:

$$\sum_{k=1}^{K} x^k(\delta_R(S)) + \sum_{k=1}^{K} (x^k + y^k)(\delta_{NR}(S)) \ge 2\left\lceil \sum_{i\in V_R\cap S} d_i/Q \right\rceil, \quad \forall S \subseteq V \setminus \{v_0\}. \tag{17}$$

Max-time inequalities are based on the limit L to the duration of a drone tour. Consider two sets $F \subseteq E_R$ and $S \subseteq V_R$ such that the duration of the optimal tour (or a lower bound to it) starting and ending at v_0, traversing all the edges in F, and visiting all the vertices in S, is greater than L. On the one hand, we have that a single drone k cannot service all the arcs in F and all the vertices in S and, hence, inequalities

$$x^k(F) + z^k(S) \le |F| + |S| - 1, \quad \forall k \in \{1, \dots K\}, \tag{18}$$

are valid for the K-GRP on G. On the other hand, under the same circumstances, at least two different drones must enter any subgraph of G that contains all edges in F and all vertices in S but does not contain the depot. Hence, if $W \subset V \setminus \{v_0\}$ is a set of vertices containing S and the vertices incident with the edges in F, the following inequalities

$$\sum_{k\in\mathbb{K}} \left(x^k(\delta_R(W)) + (x^k + y^k)(\delta_{NR}(W))\right) \ge 4, \tag{19}$$

and

$$\sum_{k'\in\mathbb{K}\setminus\{k\}} \left(x^{k'}(\delta_R(W)) + (x^{k'} + y^{k'})(\delta_{NR}(W))\right) \ge 2, \quad \forall k \in \{1, \dots K\} \tag{20}$$

are also valid for the K-GRP on G. Inequalities (18), (19), and (20) can be easily generalized to the case when the number of drones needed to service the edges in a given set F and the vertices in a given set S is greater than two.

3 | A BRANCH-AND-CUT ALGORITHM FOR THE *K*-GRP

We have implemented a branch-and-cut algorithm (B&C) for the K-GRP based on the formulation proposed in Section 2.1. The fundamental component of the B&C is a cutting-plane algorithm that incorporates separation procedures for the inequalities presented in this article.

3.1 | Separation algorithms

At each iteration of the cutting-plane procedure we use some *separation algorithms* to identify valid inequalities that are violated by the current LP fractional solution. Let $(\overline{x}^k, \overline{y}^k, \overline{z}^k)$, for $k = 1, \dots, K$, denote this fractional solution, and let $\varepsilon > 0$ be a given parameter. In what follows we describe the separation procedures used for each family of valid inequalities.

Connectivity inequalities

Although connectivity inequalities (7) can be exactly separated in polynomial time by means of max-flow computations, the procedure is very time consuming and may produce many similar violated inequalities. Therefore, we decided to use heuristic algorithms for separating them. The first one is a well-known method based on the computation of the connected components of the graph induced by the edges $e \in E$ such that $\bar{x}^k(e) + \bar{y}^k(e) > \varepsilon$, plus the depot, if necessary. Inequality (7), associated with each connected component and with the edge f in it with maximum $\bar{x}^k(f)$, is checked for violation.

The second heuristic works in a smaller graph in which the connected components of the graph induced by edges with $\bar{x}^k(e) \geq 1 - \varepsilon$ are shrunk into a single vertex each. Then, all minimum cuts between the vertex corresponding to the component containing the depot and the remaining ones are calculated and their corresponding connectivity inequalities are checked for violation.

Parity inequalities

Parity inequalities can be exactly separated in polynomial time with an algorithm based on the Padberg-Rao procedure [23] for the computation of odd minimum cuts (see also Letchford et al. [18] for a more efficient procedure). For the special case where $S = \{v\}$, the exact procedure described in Ghiani and Laporte [13] gives the set $F \subseteq \delta(v)$ associated with the most violated parity inequality.

We also use a simple and fast algorithm based on the computation of the connected components of the graph induced by the edges $e \in E$ with $\bar{x}^k(e) - \bar{y}^k(e) > \varepsilon$, plus the depot, if necessary. The same procedure as for the case $S = \{v\}$ is used here to obtain the set F corresponding to the cutset associated with each component.

***p*-connectivity inequalities**

To separate these inequalities, we first look for tight connectivity inequalities by searching among the cutsets obtained with the connectivity separation procedures. Let S be a set of vertices associated with such a cutset and assume that $v_0 \in S_0 = V \setminus S$. If the connectivity inequality is tight for drone k, we compute the connected components in the subgraph induced in $G(S)$ by the edges $e \in E(S)$ with $\bar{x}^k(e) \geq 1 - \varepsilon$. For each pair C_i, C_j of such components, $s_{ij} = 2\bar{x}^k(V_O, V_j) - 2\min\{\overline{w}^k(\alpha_i), \overline{w}^k(\alpha_j)\}$ is calculated, where V_t denotes the set of vertices in C_t and α_t is the edge e in C_t (or the vertex v in $V_t \cap V_R$) with the highest value of $\bar{x}^k(e)$ (or of $\bar{z}^k(v)$). The components that maximize s_{ij} are iteratively shrunk while s_{ij} is positive. In this way, we obtain sets $S_1, \dots, S_p$ and the associated inequality (16) is checked for violation.

Capacity inequalities

Capacity inequalities (17) can be separated heuristically by using the following simple procedure. First, we build the support graph $\overline{G} = (V, E_R \cup \overline{E}_{NR})$, where $\overline{E}_{NR}$ is defined by those nonrequired edges with $\bar{c}_e = \sum_{k\in\mathbb{K}}(x_e^k + y_e^k) > 0$. Required edges have unit capacity and edges in $\overline{E}_{NR}$ have capacity $\bar{c}_e$. For each required vertex $i \in V_R$, we compute the max flow in $\overline{G}$ from the depot to i. Let $(S : V \setminus S)$ be the associated minimum cut, with $i \in S$, and $n_v = \left\lceil \sum_{j\in V_R\cap S} d_j/Q \right\rceil$. If the max flow is less than $2n_v$, this cutset defines a violated capacity inequality (17). Whether the inequality is violated or not, we define $\mathcal{K}$ as the set of vehicles k such that $x^k(\delta_R(S)) + (x^k + y^k)(\delta_{NR}(S)) < 2$. Then, the inequality

$$\sum_{k\in\mathcal{K}} x^k(\delta_R(S)) + \sum_{k\in\mathcal{K}} (x^k + y^k)(\delta_{NR}(S)) \geq 2(n_v - K + |\mathcal{K}|), \qquad \forall S \subseteq V \setminus \{v_0\} \tag{21}$$

is valid for the K-GRP and can be more violated than the initial one.

Max-time inequalities

We separate max-time inequalities (18), (19), and (20) by using two heuristic procedures based on the ones presented in [5] for the Length Constrained K-Drones Rural Postman Problem.

The first one looks for violated inequalities (18). Let $\{e_1, e_2, \dots, e_m\}$ be a set of required edges such that $x_{e_1}^k \geq x_{e_2}^k \geq \dots \geq x_{e_m}^k \geq 0.5$, and let $F = \{e_1, e_2, \dots, e_f\}$, where f is the maximal number such that $x^k(F) > |F| - 0.5$. Then we define S as the set of vertices with $z_i^k = 1$. Now we solve the GRP with the set of required edges F and the set of required vertices $S \cup \{v_0\}$ with the branch-and-cut algorithm described in [8], and let $C(F, S)$ be its optimal value or a lower bound. If $C(F, S) > L$ the corresponding inequality (18) is violated. Otherwise, for each edge $\bar{e} \in \{e_{f+1}, \dots, e_m\}$, we consider the set $\overline{F} = F \cup \{\bar{e}\}$ and check if $C(\overline{F}, S)$ is greater than L. For each pair (F, S) (or $(\overline{F}, S)$) whose corresponding inequality (18) is violated, we look for the cutset of minimum weight between the depot and the edges in F and vertices in S in the support graph $\overline{G} = (V, E_R \cup \overline{E}_{NR})$ defined above for the separation of capacity inequalities. For this cutset, we check the corresponding max-time inequalities (19) and (20) for violation.

The second heuristic looks for inequalities (19). The procedure starts by defining $S = \{i\}$, where i is the vertex farthest from the depot such that the maximum flow from v_0 to i in $\overline{G}$ is less than $2K$. Then we iteratively add vertices to S in such a way that $\sum_{k=1}^K x^k(\delta_R(S)) + \sum_{k=1}^K (x^k + y^k)(\delta_{NR}(S))$ is minimum. For each S, we compute the minimum number of vehicles needed to service all the edges in $E_R(S)$ and all the required vertices in S by solving the associated GRP. The corresponding inequality (19) is checked for violation. If a violated constraint (19) is found, at least one of the inequalities (20) is also violated and it is added.

3.2 | The cutting-plane algorithm

The initial LP relaxation contains inequalities (3), (4), (5), (6), (8), (9), and the bounds on the variables. Moreover, as is usual in routing problems with several vehicles, some symmetry-breaking inequalities are added to avoid equivalent solutions.

At each iteration, the cutting-plane algorithm applies the first heuristic algorithm for connectivity inequalities with $\varepsilon = 0, 0.25, 0.5$, where the value of ε is increased only if no violated inequalities are found with the previous value. If this heuristic fails in finding violated cuts, the second connectivity heuristic is applied for $\varepsilon = 0, 0.1, 0.2, 0.3, 0.4$. All the tight cut-sets obtained with the connectivity separation procedures are stored and used by the p-connectivity heuristic to check the violation of their associated inequalities with $\varepsilon = 0, 0.15, 0.3$.

Moreover, at each iteration we apply the heuristic for identifying violated capacity inequalities, the exact parity separation procedure for single vertices, and the heuristic procedure for parity inequalities with values 0,0.25, and 0.5 for ε. A value of ε is used only if the previous one results in no violated inequalities.

Max-time separation algorithms are called only at the root node. Furthermore the exact procedure for parity inequalities is applied only at the root node if no violated connectivity, p-connectivity, nor parity inequalities have been found so far.

4 | A MATHEURISTIC FOR THE MULTI-PURPOSE K-DRONES GENERAL ROUTING PROBLEM

In this section we present a matheuristic algorithm for the MP K-DGRP that consists of two parts. The first part aims to find solutions for the K-GRP considering that each original line of an instance is represented by a single required edge (without intermediate points). Let us call this instance K-GRP(0). In this part, we use an order-first split-second method (see [25]) that initially generates a "giant tour" traversing all the required edges and visiting all the required nodes of G without considering the max-time and capacity constraints. Then, K feasible drone routes for the K-GRP(0) instance are obtained from it. This is described in Section 4.1. Several different giant tours are generated and partitioned into K routes to obtain a set of K-GRP(0) feasible solutions. These initial solutions are improved by applying a variable neighborhood descent (VND) algorithm, described in Section 4.2, that combines four local search procedures and a route optimization phase.

The second part of the matheuristic is focused on improving the n best K-GRP(0) solutions obtained in the previous part by adding some intermediate points to the required edges, thus allowing drones to enter (or to exit) the lines not only at its endpoints, but also at a subset of intermediate points. First we consider the new instance K-GRP(1) resulting from adding one intermediate vertex to each required edge, so that each line is approximated by a polygonal chain with two segments (edges) with the same traversal (and service) time. Each one of the n K-GRP(0) solutions previously obtained is trivially transformed into a K-GRP(1) solution. We call this procedure "1-splitting". We then apply again the VND algorithm and the route optimization procedure to each K-GRP(1) solution. Some of these solutions may have been now improved due to drones entering or leaving some required edges through its middle point. The most "promising" edges of each solution, those whose two halves are now serviced by different drones (or by the same drone but not consecutively), are split again by adding p equidistant intermediate vertices. We then try to improve the solution by using these new vertices. This is detailed in Section 4.3.

Throughout this section, we will use K-GRP *solution* to refer to a set $\mathcal{T} = (T_1, T_2, \dots, T_K)$ of K routes, each one starting and ending at the depot, each with duration no greater than L and total demand not exceeding the drone capacity Q, such that each required edge is traversed and each required node is visited. We will use *task* $t_i = (t_{i1}, t_{i2})$ to refer either to a required edge (t_{i1}, t_{i2}) serviced by traversing it from t_{i1} to t_{i2} or to a required vertex $t_{i1} = t_{i2}$. Then, a route associated with drone k can be represented by a sequence of tasks $T_k = \{t_i^k, \dots, t_j^k\}$, where it is assumed that the deadheading from the depot to the first task, from the end of a task to the beginning of the following one, and from the last task back to the depot, is done by traversing the corresponding nonrequired edge. We will denote by $d(t_\ell^k)$ the demand of task $t_\ell^k, t_\ell^k \in V_R \cap T_k$. A route will be *feasible* if its duration is not greater than L and if its total demand does not exceed Q.

4.1 | Solutions for K-GRP(0)

As mentioned above, this first part of the matheuristic focuses on finding solutions for the K-GRP(0) instance. The algorithm starts by finding an optimal tour on $G = (V, E)$ traversing all the required edges and visiting all the required nodes of G. This tour, commonly called a *giant tour* in the literature, is generated by relaxing drone capacity and time limit, and solving the GRP instance optimally with the branch-and-cut algorithm described in [8]. The tour returned by this procedure has an associated sequence of tasks T_G.

The giant tour is optimally partitioned into K feasible drone routes by solving a shortest path problem over an auxiliary directed graph $G^* = (V^*, A^*)$ (see [3, 31]). This procedure was already used in [5] and has been extended to consider required vertices as follows:

The set V^* contains $|V_R| + |E_R| + 1$ nodes, where v_0 denotes the depot and the remaining nodes v_ℓ represent the tasks t_ℓ. The nodes are arranged from left to right following the order in which their associated tasks are performed in the giant tour. Each arc in A^* represents a feasible drone route on G. An arc from node v_i to node v_j is added if the route starting from the

Algorithm 1. Splitting procedure for a giant tour T_G

```
Require: $G, T_G$
1: initialize $G^* = (V^*, A^*)$: $V^* \leftarrow \{0, 1, \dots, n\}$; $A^* \leftarrow \emptyset$
2: for $i \leftarrow 1$ to $n$ do
3:   $j \leftarrow i$
4:   $arctime \leftarrow 0$
5:   $demand \leftarrow d(t_j)$
6:   while ($j \leq n$) and ($demand \leq Q$) and ($arctime \leq L$) do
7:     $arctime \leftarrow \gamma(i-1, j)$
8:     if $arctime \leq L$ then
9:       add arc $(i-1, j)$ to $A^*$ with traversal time $arctime$
10:      $j \leftarrow j+1$
11:      $demand \leftarrow demand + d(t_j)$
12:    end if
13:  end while
14: end for
15: compute the shortest path $P$ from 0 to $n$ in $G^*$
16: transform each arc of $P$ into a sequence of ordered tasks
17: return a $K$-GRP(0) solution (a set of $K$ drone routes)
```

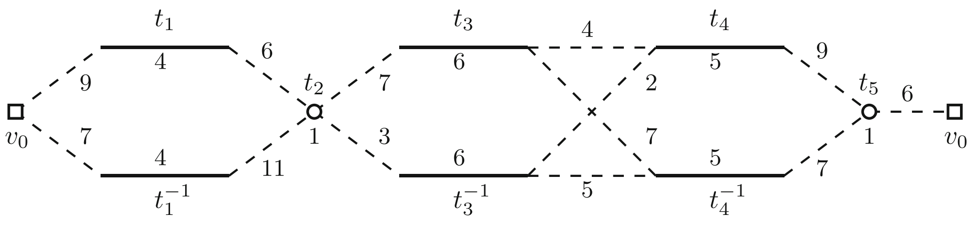


FIGURE 3 Example of an auxiliary graph used in the *Split with Flips* procedure.

depot, performing tasks $t_{i+1}, \dots, t_j$ in that order, and going back to the depot, is feasible. The time associated with these arcs, denoted $\gamma(i,j)$, is the duration of the corresponding route.

In graph G^* we compute a shortest path from node v_0 to node $v_{|V_R|+|E_R|}$, whose set of arcs, as proved in Ulusoy [31], defines a partition of the giant tour into K feasible tours that is optimal regarding the ordering of the traversal of the tasks. Algorithm 1 summarizes this procedure. However, this method does not take into account that required edges can be traversed in two possible directions. Prins et al. [24] propose a procedure called *Split with Flips* that is a generalization of the previous method considering the two directions in which each edge can be traversed. The main difference consists of the way the times $\gamma(i,j)$ associated with the arcs in G^* are calculated.

We have adapted this algorithm to consider required vertices as follows. For each arc in $a \in A^*$ representing a route, a new auxiliary directed graph G_a^* is created in order to calculate its associated duration. Each task corresponding to a required edge in G is represented now with two arcs, one for each possible direction of traversal, with times equal to its original service time. For each task associated with a required vertex, one vertex is created with a (visiting) time equal to its original service time in G. Then an arc is added from the end vertex of each arc (or vertex) representing a task to the initial vertex of the following arc (or vertex), whose time is that of the corresponding shortest path in G. Two additional vertices representing the depot are added and connected with the first and last task of the route, respectively, with times equal to those of the corresponding shortest paths in G. In this graph, a shortest path between the two copies of the depot is calculated. This shortest path determines the optimal direction of traversal of each task and its time will be the one assigned to arc $a \in A^*$.

Figure 3 illustrates the graph generated for a subsequence $T = \{t_1, t_2, t_3, t_4, t_5\}$ of tasks, where arc t_ℓ represents task t_ℓ traversed in the direction given by the route and arc t_ℓ^{-1} corresponds to the opposite direction of traversal. The numbers next to the nodes and edges represent their service times. Dashed lines represent the nonrequired edges corresponding to the shortest paths joining the tasks. In this example, the shortest path $\{t_1, t_2, t_3^{-1}, t_4, t_5\}$ has value 52, while the directions given by the giant tour have a value of 58.

In order to obtain a larger set of initial solutions, we apply the above algorithm with other initial giant tours on G defined by different Eulerian circuits obtained from the optimal GRP solution. Note that a GRP solution is an Eulerian graph that

Algorithm 2. Pseudocode of the VND algorithm

```
1:  for each $K$-GRP(0) solution $\mathcal{T} \in \bar{\mathcal{T}}$ do
2:      $\rho \leftarrow 1$
3:      while $\rho \leq 4$ do
4:          obtain $\mathcal{T}' \in \mathcal{N}_\rho(\mathcal{T})$ by applying the $\rho$-th local search method to $\mathcal{T}$
5:          if $f(\mathcal{T}') < f(\mathcal{T})$ then
6:              $\mathcal{T} \leftarrow \mathcal{T}'$ and $\rho \leftarrow 1$
7:          else
8:              $\rho \leftarrow \rho + 1$
9:          end if
10:     end while
11: end for
```

can be traversed in different ways. We try to generate different Eulerian circuits by applying the Hierholzer algorithm [15] $|E_R| + |V_R|$ times, starting each time with a different task. We thus obtain a set $\overline{\mathcal{T}}$ of different initial K-GRP(0) solutions, with $|\overline{\mathcal{T}}| \leq |E_R| + |V_R|$.

4.2 | A variable neighborhood descent algorithm for the K-GRP(0)

Once the initial set of K-GRP(0) feasible solutions is generated, a variable neighborhood descent algorithm (VND) [11, 20] is applied to each solution $\mathcal{T} \in \overline{\mathcal{T}}$ to try to improve it. A VND is a metaheuristic that explores a sequence $\mathcal{N} = (N_1, \ldots, N_{\rho_{\max}})$ of neighborhood structures in a deterministic way. Starting from an initial solution $\mathcal{T}$ and $\rho = 1$, each iteration of the VND explores the neighborhood $N_\rho(\mathcal{T})$ to try finding a better solution. If one improving move is detected, it is executed and ρ is reset to 1; otherwise, ρ is incremented to browse the next neighborhood. The algorithm stops when the exploration of the last neighborhood $N_{\rho_{\max}}(\mathcal{T})$ brings no improvement, that is, when the current solution $\mathcal{T}$ is a local optimum over all the considered neighborhoods. The VND algorithm proposed here explores $\rho_{\max} = 4$ different neighborhood structures (see Algorithm 2).

The first neighborhood structure N_1 is defined by the *intra-route* move and consists of all the permutations resulting from moving a task to another position within the route that performs it. For each route T_k of a solution $\mathcal{T}$, the intra-route procedure removes a task $t_\ell \in T_k$ at each step and inserts it in the position of T_k that minimizes the duration of the route.

The second neighborhood structure N_2 is defined by the *destroy and repair* move. Each iteration of the associated local-search procedure randomly chooses r tasks, with $2 \leq r \leq 8$, and removes them from their corresponding routes. Then, the algorithm tries to relocate each task one by one in the route and position that minimizes the total time, satisfying the time limit and the capacity constraints. Note that it is possible that a required edge can not be placed in its original position because another edge has been previously added to its route. If an edge cannot be inserted in any route, a new route servicing it is created. If the total time of the new solution is not better than the time of the original one, the changes made in this iteration are discarded. This procedure is repeated until one improving move is detected or until $i_{\max}$ consecutive iterations without any improvement are performed, with $i_{\max}$ a given parameter.

The third neighborhood structure N_3 is defined by the 0 to ℓ-exchange move. Each iteration of this local-search procedure removes ℓ consecutive tasks from the route servicing them and inserts all of them between two consecutive tasks of another route. The algorithm considers the removal of all the possible sets of ℓ consecutive tasks and their insertion in all the possible positions of other routes such that the duration and the total demand of the resulting route do not exceed L and Q, respectively. The algorithm starts with $\ell = 1$, and if no exchange improves the original solution, ℓ is incremented by 1 and the process is repeated. The procedure stops when an improving move is executed, or when $\ell = \ell_{\max}$ and there are no moves that improve the total time, with $\ell_{\max}$ a given parameter.

The last neighborhood structure N_4 is defined by the *ℓ_1 to ℓ_2-exchange* move and contains all the solutions obtained by interchanging a chain of ℓ_1 consecutive tasks from one route with a chain of ℓ_2 consecutive tasks from another route, with $\ell_1 \leq \ell_2$. The local-search procedure starts with $\ell_1 = \ell_2 = 1$, and tries to interchange the tasks between the two routes in order to find an improving move. If no exchange move reduces the total time satisfying the route capacity and time limit constraints, ℓ_2 is incremented by one unit and the process is repeated. If ℓ_2 reaches $\ell_{\max}$ and no improving exchanges are found, then ℓ_1 increases by one unit and ℓ_2 is set equal to ℓ_1. The procedure stops when an improving move is found, or if $\ell_1 = \ell_2 = \ell_{\max}$ and there are no exchange moves that produce a better solution.

Once the VND algorithm is terminated, a *route-optimization* procedure is applied to each solution $\mathcal{T} \in \overline{\mathcal{T}}$. For each drone route T of a solution $\mathcal{T}$, this procedure defines a GRP instance on graph G formed by the required edges and the required nodes

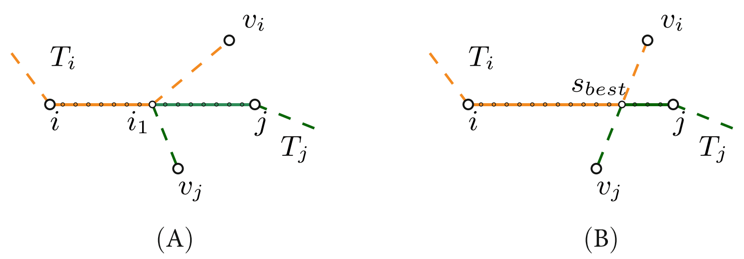


FIGURE 4 Illustration of p-splitting. (A) Adding p new intermediate points, (B) selecting the best intermediate point.

corresponding to the tasks performed on T. This GRP instance is then optimally solved with the branch-and-cut algorithm proposed in [8]. The resulting routes will have a total time less than or equal to that of the original ones.

4.3 | Adding intermediate points to obtain better K-GRP solutions

Let $\overline{\mathcal{T}}_b$ be the subset of the n best K-GRP(0) solutions obtained in the first part of the matheuristic. We add an intermediate vertex i_1 (equidistant from both endpoints) to each required edge (i,j) of G to obtain a new instance called K-GRP(1) with two edges $(i,i_1),(i_1,j)$ for each original required edge (i,j). Given a solution $\mathcal{T} \in \overline{\mathcal{T}}_b$ of K-GRP(0), it is easy to transform it into a solution $\mathcal{T}'$ of K-GRP(1) with the same total duration that traverses edges $(i,i_1),(i_1,j)$ consecutively.

Let $\overline{\mathcal{T}}_1$ be the set of all the K-GRP(1) solutions obtained in this way. The VND algorithm and the route optimization procedure are applied to each $\mathcal{T}' \in \overline{\mathcal{T}}_1$ to try to improve their overall duration. Observe that drones can now enter and leave any line through its middle point, which may lead to a better solution where the service of some original lines can be shared by two drones.

Let $E_1^{\mathcal{T}'}$ be the set of original lines whose middle point is incident to nonrequired edges in solution $\mathcal{T}'$. Note that an edge in $E_1^{\mathcal{T}'}$ is serviced by two drones (or by the same drone but not servicing both halves consecutively). It may be possible to improve this solution by "moving" this middle point closer to the extreme points of the edge. To do this, we consider p, with p odd, intermediate vertices evenly spaced in the line. Note that the middle point is one of them. Then, we study the improvement obtained by changing the entry/leaving point for this edge to each one of the $p-1$ other new points. This procedure, which we call "p-splitting", is applied to all the edges in $E_1^{\mathcal{T}'}$.

Figure 4A illustrates how the "p-splitting" procedure improves a solution where two (not necessarily different) routes, T_i (in orange) and T_j (in green), are involved in servicing an original required line joining vertices i and j. Note that these routes enter or leave the line at its middle point i_1 and, thus, i_1 is incident with two nonrequired edges (dashed lines) in the solution. After analyzing the total duration of the routes obtained by replacing vertex i_1 by all the p intermediate points, the best solution is the one that uses vertex s_{best}, which is depicted in Figure 4B.

This p-splitting procedure is applied with $p = 15$ to each solution $\mathcal{T}' \in \overline{\mathcal{T}}_1$ and the best solution obtained is selected as the final solution of the matheuristic.

Figure 5A,B illustrates an example of a solution before and after applying the procedure described in this subsection. For this instance, the deadheading time is reduced by 5.2% due to drones entering and leaving six of the original required edges through some of their intermediate points.

5 | COMPUTATIONAL EXPERIMENTS

In this section, we present the instances we have generated to analyze the behavior of the proposed matheuristic and branch-and-cut algorithms, as well as the computational study performed. The algorithms have been implemented in C++ and all the tests have been run on an Intel Core i7 at 2.8 GHz with 16 GB RAM. The B&C uses CPLEX 12.10 MIP Solver with a single thread. CPLEX heuristic algorithms were turned off, and CPLEX's own cuts were activated in automatic mode. The optimality gap tolerance was set to zero and best bound strategy was selected. The branch-and-cut algorithm used for obtaining the initial optimal giant tour and for optimizing the routes after each VND improvement phase was also coded in C++ and uses CPLEX 12.10 MIP Solver too.

5.1 | Instances

As defined above, a multi-purpose K-Drones GRP instance is given by several sets of parallel lines (each set covering a continuous area), some isolated vertices for delivery (those required nodes with positive demand), and a depot. In order to evaluate

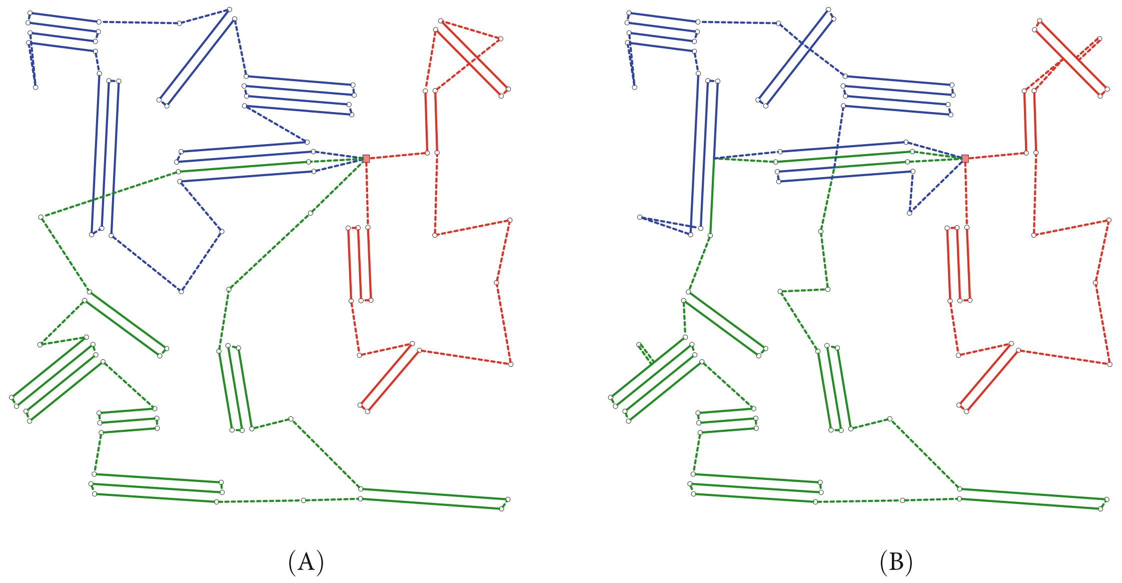


FIGURE 5 Two solutions of instance MPDGRP882 before and after applying the splitting part of the matheuristic. Deadheading time: (A) 4341.9 and (B) 4116.27.

the behavior of the branch and cut and the matheuristic algorithm, we have randomly generated two different types of instances, both differing in the way they are built, classified as Type I or Type II instances.

First, we select values for n and m, and generate a GRP instance on a grid with $n \times m$ points whose coordinates are multiples of 100. The graph initially contains all the vertical and horizontal edges, as well as some random diagonals. For type I instances generation, we divide the grid into a number of regions computed by $nareas = \lceil \frac{n}{3} \rceil \cdot \lceil \frac{n}{3} \rceil$ and we randomly declare required an edge in each region. Each one of these required edges gives rise to a required area later. This procedure ensures that areas are disjoint and homogeneously distributed throughout the grid.

For type II instances generation, each edge of the original grid is considered required (and therefore included on the instance) with probability p, thus obtaining several connected components of required edges. As some of these components may contain more than one edge, we iteratively remove required edges that are incident to vertices with degree greater than one (at random) until we only have isolated required edges. The idea behind this second type is to generate larger instances with a set of required areas that are closer to each other. These type of instances can also be more difficult for our algorithms.

Once a representative required line from each area is generated, the following steps are identical for both types of instance. We randomly select the depot and a number $nvreq$ of isolated required nodes among those vertices of the grid that are not incident with the selected required edges. The vertices, except for the depot, that are not required nor incident with required edges are removed, and the coordinates of all the vertices of the instance are randomly perturbed by adding a value in $[-20, 20]$ to each coordinate, avoiding completely horizontal and vertical edges and also slightly changing the length of the edges. Each required area is completely defined by adding a set of $npar(e)$ parallel lines to each initial required edge e (separated by a distance $distpar$ to each other). The length and position of these new edges are then slightly perturbed in a random way so that the represented area has a more irregular shape. The demand of each required node v is given by a value $dem(v)$ and the service time of each required edge is the Euclidean distance multiplied by a parameter $timefactor$.

We have generated MP K-DGRP instances with different values for $n, m \in \{6, 7, 8, 9, 10, 12\}$, $npar(e) \in \{1, 2, 3\}$ for each initial required edge e, $distpar = 20$ and $timefactor = 1.5$. For type I instances, we have generated two sets with $nvreq \in [n-4, n+4]$: one with $dem(v) \in \{1, 2, 3\}$ for each required node v (version 1), and another with unit demands (version 2). Since the demands of the vertices represent the weight of the delivery, and given that it seems reasonable that the total weight that can be carried by a drone is not very large, we have not considered demands greater than 3. For type II instances, we have also generated two different sets with unit demands: one with $p = 0.4$ and $nvreq \in [n-1, n+1]$ (version 1), and another with $p = 0.3$ and $nvreq \in [2n-1, 2n+1]$ (version 2).

The characteristics of all the MP K-DGRP instances generated are shown in Table 1 and can be found in http://www.uv.es/plani/instancias.htm. Table 1a shows, for each MP K-GRP instance of type I, the number of vertices, the number of required vertices and its total demand, the number of required lines, and the number of areas. Table 1b shows the type II instances characteristics and presents the same structure except for the total demand, which is not included because its value matches the number of required vertices. The digits in the name of each instance indicate the values of m, n and if this is the first or the second version generated from the same grid. Examples of a type I instance, a type II instance (version 1), and a type II instance (version 2) on a grid with $n = m = 9$ are shown in Figure 6A–C.

TABLE 1 Characteristics of the MP K-DGRP instances

(a) Type I instances

Instance name	$\|V\|$	$\|V_R\|$	D	$\|E_R\|$	♯ areas
MPDGRP661	31	6	14	12	4
MPDGRP662	37	6	6	15	4
MPDGRP681	48	7	16	20	6
MPDGRP682	50	7	7	21	6
MPDGRP771	32	5	10	13	4
MPDGRP772	34	5	5	14	4
MPDGRP861	50	7	15	21	6
MPDGRP862	55	8	8	23	6
MPDGRP881	32	5	10	13	4
MPDGRP882	39	6	6	16	4
MPDGRP8101	79	12	24	33	12
MPDGRP8102	93	10	10	41	12
MPDGRP991	61	10	22	25	9
MPDGRP992	69	10	10	29	9
MPDGRP1081	83	12	24	35	12
MPDGRP1082	85	12	12	36	12
MPDGRP10101	61	10	22	25	9
MPDGRP10102	66	11	11	27	9
MPDGRP12121	103	10	22	46	16
MPDGRP12122	111	12	12	49	16

(b) Type II instances

Instance name	$\|V\|$	$\|V_R\|$	$\|E_R\|$	♯ areas
MPDGRP661	60	7	26	8
MPDGRP662	54	13	20	6
MPDGRP681	83	6	38	11
MPDGRP682	70	13	28	9
MPDGRP6101	114	7	53	17
MPDGRP6102	85	12	36	14
MPDGRP771	105	8	48	14
MPDGRP772	71	14	28	9
MPDGRP791	134	7	63	20
MPDGRP792	81	14	33	11
MPDGRP881	118	9	54	18
MPDGRP882	103	16	43	15
MPDGRP8101	150	9	70	24
MPDGRP8102	97	16	40	12
MPDGRP991	157	10	73	25
MPDGRP992	74	19	27	11
MPDGRP9101	163	10	76	23
MPDGRP9102	94	19	37	14
MPDGRP10101	179	10	84	28
MPDGRP10102	143	20	61	19

In order to choose the values for L, each instance has been executed several times with different L values to guarantee that the solutions will use a number of drones ranging from 2 to 6. The capacity Q of the drones has been chosen so that all the demand of the required vertices can be serviced while keeping the number of packages carried by each drone as low as possible.

5.2 | Computational results

We present here the results obtained with the branch-and-cut and the matheuristic algorithms on the MP K-DGRP instances. First we have applied the B&C algorithm on the K-GRP(0) instances, that is, on the instances without adding any additional

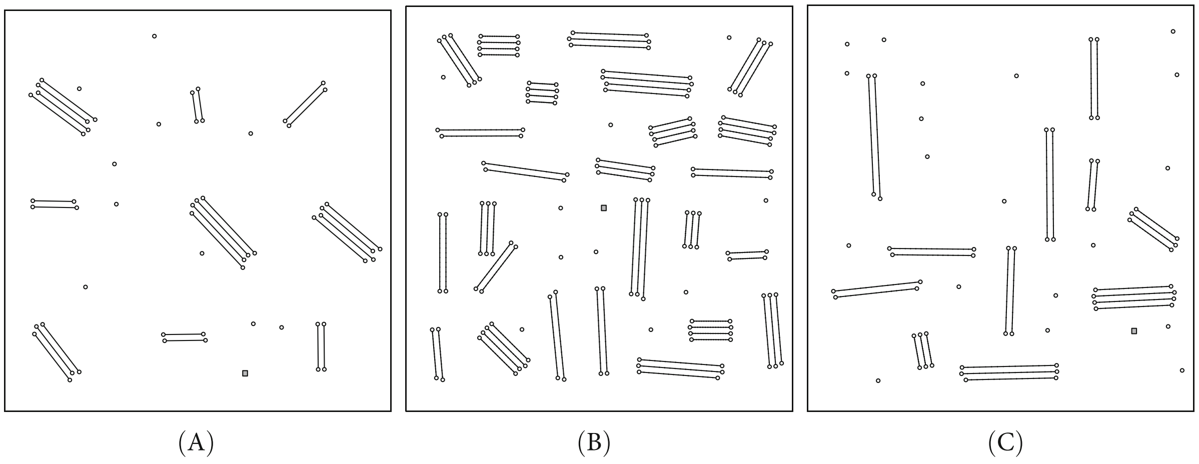


FIGURE 6 MP K-DGRP instances on a grid 9×9. (A) Type I, (B) Type II (Version 1), (C) (B) Type II (Version 2).

TABLE 2 Computational results with the B&C for K-GRP(0) and the matheuristic on the instances of Type I.

			Branch-and-Cut					Matheuristic				
	$\|V\|$	K	♯ opt	Gap0 (%)	Gap (%)	Nodes	Time	♯ opt	♯ best	Gap (%)	Imp (%)	Time
General Demands	≤ 60	2	5	3.86	0.00	57.0	3.9	5/5	5	0.00	0.19	3.4
		3	5	5.82	0.00	178.2	16.3	4/5	4	0.13	0.02	3.9
		4	5	8.78	0.00	237.2	24.8	5/5	5	0.00	0.11	2.9
		5	5	12.72	0.00	654.6	98.5	4/5	4	0.00	0.67	3.7
		6	5	17.02	0.00	2590.8	427.0	3/4	3	0.13	0.65	4.4
	> 60	2	5	3.00	0.00	199.0	49.1	4/5	4	0.00	0.15	30.1
		3	5	5.22	0.00	705.6	349.9	2/5	2	0.09	0.02	19.9
		4	3	8.80	0.82	5498.6	3828.4	2/3	4	0.98	0.26	19.9
		5	2	10.07	2.46	4224.0	5501.0	1/2	3	2.67	0.57	17.8
		6	0	11.24	4.50	5042.6	7200.0	0/0	3	4.97	0.20	17.4
Unit Demands	≤ 60	2	5	3.17	0.00	46.2	6.3	4/5	4	0.01	0.01	4.8
		3	5	6.47	0.00	206.4	25.5	5/5	5	0.00	0.01	4.1
		4	5	10.75	0.00	348.8	60.2	3/5	3	1.28	0.10	4.2
		5	5	15.17	0.00	1991.4	1511.9	5/5	5	0.00	0.26	4.5
		6	5	17.74	0.00	4065.2	1736.5	4/5	4	0.02	0.83	4.4
	> 60	2	5	3.95	0.00	608.8	112.7	3/5	3	0.28	0.11	43.7
		3	5	5.11	0.00	381.0	148.7	3/5	3	0.21	0.02	34.4
		4	3	8.34	1.02	2899.4	3211.5	1/3	3	1.04	0.08	28.0
		5	1	12.01	2.73	4659.8	5692.9	1/1	4	2.83	0.28	26.9
		6	0	13.18	5.03	4196.2	7200.0	0/0	5	5.03	0.24	27.5
Total			**79**					**59/79**	**76**			

intermediate vertices. Later, we will study the behavior of the B&C in the Type I instances to whose required edges we will add an additional intermediate vertex to get their K-GRP(1) associated instances. Each of these 40 instances, 20 of type I and 20 of type II, is solved with 2, 3, 4, 5 and 6 drones, and, hence, a total of 200 instances have been run.

Tables 2 and 3 summarize the computational results obtained with both algorithms for all the instances of type I and type II, respectively. Both tables present the same structure. The results for each type of instance are separated in two blocks by a double horizontal line according to the instance generation characteristics (general or unit demands in type I instances, and version 1 or version 2 in type II instances). Each of these blocks is also separated according to the number of vertices. The number of drones used by the solution is shown in Column 2. Each row refers to the data of 5 instances.

The results obtained with the B&C algorithm for the corresponding K-GRP(0) instances with a time limit of 7200 seconds are reported in columns 3 to 7. Column 3 shows the number of instances out of five solved to optimality. Columns 4 and 5 show the average percentage gaps between the value of the optimal solution (or the best upper bound found) and the lower bound at the end of the root node ("Gap0") and the final lower bound ("Gap"), respectively. Column 6 reports the average number of nodes of the branching tree and Column 7 shows the average total time, in seconds, used by the B&C.

TABLE 3 Computational results with the B&C for K-GRP(0) and the matheuristic on the instances of Type II.

			Branch-and-Cut					Matheuristic				
	$\|V\|$	K	♯ opt	Gap0 (%)	Gap (%)	Nodes	Time	♯ opt	♯ best	Gap (%)	Imp (%)	Time
Version 1	≤ 120	2	5	2.69	0.00	2775.2	761.6	5/5	5	0.00	0.30	68.2
		3	4	4.08	0.07	5702.8	4240.7	1/4	2	0.15	0.38	58.0
		4	2	5.00	1.48	4610.2	5792.5	2/2	5	1.48	0.48	46.0
		5	0	7.35	4.56	3594.0	7200.0	0/0	5	4.56	0.69	44.3
		6	0	8.58	6.02	2892.4	7200.0	0/0	5	6.02	0.66	39.7
	> 120	2	2	2.01	0.33	3746.2	4750.6	1/2	2	0.51	0.26	285.0
		3	0	3.03	1.66	2535.4	7200.0	0/0	2	1.83	0.41	215.4
		4	0	4.02	2.80	1303.2	7200.0	0/0	5	2.80	0.18	170.0
		5	0	5.45	4.51	814.2	7200.0	0/0	4	4.51	0.25	162.0
		6	0	7.46	6.77	499.0	7200.0	0/0	5	6.77	0.45	154.3
Version 2	≤ 84	2	5	2.86	0.00	491.2	44.8	2/5	2	0.04	0.38	24.5
		3	5	4.71	0.00	2947.2	1521.3	2/5	2	0.23	0.39	19.3
		4	3	5.99	0.90	4334.0	4362.8	1/3	3	1.19	0.84	19.2
		5	1	9.21	3.70	3694.6	7200.0	1/1	5	3.70	0.84	18.6
		6	0	10.67	6.07	2545.0	7200.0	0/0	5	6.07	0.71	19.0
	> 84	2	5	2.29	0.00	1144.6	283.9	1/5	1	0.40	0.44	130.9
		3	3	4.13	0.44	4166.2	4446.0	1/3	3	0.57	0.43	82.6
		4	0	6.28	3.47	2591.8	7200.0	0/0	4	3.55	0.40	89.3
		5	0	9.41	6.99	1504.0	7200.0	0/0	5	6.99	0.52	71.9
		6	0	11.04	8.94	1017.2	7200.0	0/0	5	8.94	0.20	70.3
Total			**35**					**17/35**	**75**			

From Table 2 we observe that the B&C is capable of solving all type I instances with up to 60 vertices and 2, 3, 4, 5, and 6 drones in less than 30 minutes on average of computing time. Regarding the instances with more than 60 vertices, it can be seen that most of the instances with 2, 3, and 4 drones (26 out of 30) have been solved in less than one hour of computing time. However, the B&C has only been able to solve 3 out of 20 instances with 5 and 6 drones. On the other hand, comparing the values of "Gap0", "Gap", and "Time" obtained for the instances with general demands and unitary demands, we can observe that the latter seem a bit more difficult for the B&C than those with general demands. This apparently small increase in difficulty could be explained by the use of CPLEX cover inequalities in the case of general demands.

Table 3 reports the results obtained by the B&C and the matheuristic in Type II instances, which are associated with larger graphs. It can be seen that only a small number of instances have been solved to optimality. Specifically, 17 instances out of 20 with 2 drones, 12 out of 20 with 3 drones, and 6 with 4, 5, and 6 drones out of 60 instances. Note, however, that except for the largest instances with 6 drones, the final gaps obtained are very good, showing that the inequalities in the formulation and the proposed valid inequalities provide a good description of the convex hull of the solutions of the problem. These results reflect the great difficulty of the problem when the instance is large and encourage the development of heuristic algorithms for solving the multi-purpose K-drones general routing problem.

The results obtained with the matheuristic on all the instances of type I and II are reported in columns 8 to 12 of Tables 2 and 3. To analyze its performance, we have compared the results of the first part of the matheuristic (without adding intermediate points) with those of the branch and cut on the K-GRP(0) instances. Note that the final solutions of the matheuristic (provided after the second part of the algorithm) are not feasible solutions of the K-GRP(0) instance, but we compare them with the solutions obtained in the first part to be able to analyze the improvement of allowing drones to use some intermediate vertices.

Column 8 shows the number of optimal solutions of the K-GRP(0) instances found with the matheuristic, and column 9 reports the number of times the matheuristic reaches the optimal solution or provides the best upper bound. The "Gap" column shows the average percentage gap between the value of the solution provided by the first part of the matheuristic and the lower bound given by the branch-and-cut algorithm in two hours of computing time. Column "Imp" gives the average percentage improvement after applying the second part of the matheuristic algorithm with respect to the solution provided in the first part. The last column reports the average total computing time, in seconds, used by the matheuristic.

We can see that in 151 out of the 200 instances considered, the matheuristic provides a K-GRP(0) solution equal to or better than the one obtained with the B&C in a much more reasonable computing time. Nearly 67 % of the K-GRP(0) instances optimally solved by the B&C are also optimally solved by the matheuristic.

TABLE 4 Results for the instances with known optimal solutions.

Type	K	♯ opt B&C	♯ opt M	Gap (%)	Gap-p (%)	Time
I	2	20	16	0.07	−0.04	20.5
II		17	9	0.15	−0.23	95.9
I	3	20	14	0.11	0.09	15.6
II		12	4	0.18	−0.27	51.1
I	4	16	11	0.47	0.33	9.8
II		5	3	0.29	−0.55	24.4
I	5	13	11	0.00	−0.37	6.0
II		1	1	0.00	−0.56	9.0
I	6	10	7	0.08	−0.69	4.4
II		0	-	-	-	-
Total		**114**	**76**			

TABLE 5 Results with the B&C on the instances of Type I for K-GRP(1).

	$\|V\|$	K	♯ opt	Gap0 (%)	Gap (%)	Nodes	Time	Imp (%)
General Demands	≤ 60	2	5	4.62	0.00	54.0	9.9	0.09
		3	5	6.94	0.00	360.4	102.5	0.01
		4	5	8.48	0.00	1512.6	627.5	0.18
		5	4	12.62	1.07	1600.6	1827.9	1.27
		6	3	16.85	1.79	4774.8	3571.3	1.29
	> 60	2	5	3.24	0.00	508.6	237.8	0.08
		3	4	5.35	0.19	1003.8	1857.7	0.00
		4	1	9.00	3.07	2252.4	5985.3	0.07
		5	0	10.34	4.39	1826.4	7200.0	0.53
		6	0	12.15	8.09	1099.2	7200.0	0.11
Unit Demands	≤ 60	2	5	3.85	0.00	67.8	14.2	0.00
		3	5	6.00	0.00	305.0	41.1	1.32
		4	5	10.78	0.00	1892.8	447.7	0.02
		5	3	15.63	3.25	2430.2	3210.3	0.26
		6	1	18.16	5.58	7215.6	6244.4	0.68
	> 60	2	5	3.82	0.00	867.8	675.2	0.05
		3	5	5.20	0.00	713.8	749.5	0.00
		4	2	8.97	3.25	1213.4	4908.2	0.24
		5	0	12.67	7.86	925.8	7200.0	0.16
		6	0	14.69	11.47	604.2	7200.0	0.17
Total			**63**					

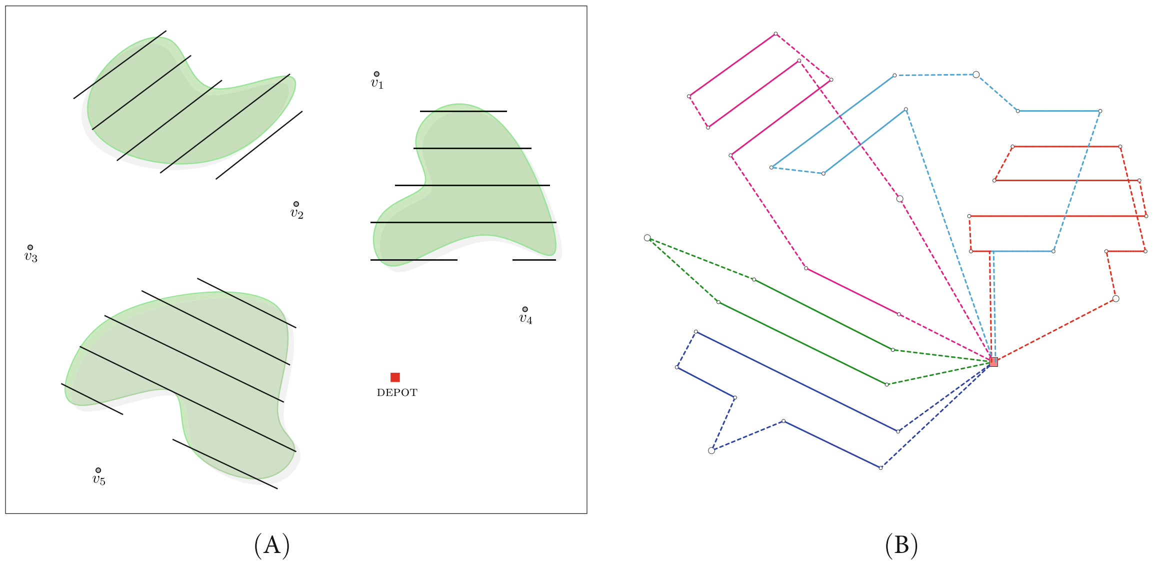
Table 4 summarizes the results obtained by the matheuristic on the 114 instances for which an optimal solution is known for the corresponding K-GRP(0) instance. Columns 1 and 2 contain the instance type and the number of drones. Column 3 reports the number of K-GRP(0) instances with known optimal value, and Column 4 the number among them for which the first part of the matheuristic provides the optimal solution. The "Gap" column shows the average percentage gap between the value of the solution of the first part of the matheuristic and the optimal solution, while "Gap-p" represents the same gap for the solution obtained by the matheuristic after the p-splitting phase. Note that some of these latter gaps may be negative, since the solutions provided by the matheuristic for the instance with intermediate vertices may be better than the optimal solution of the K-GRP(0) instance. The "Time" column reports the average computing time in seconds.

As mentioned above, the first part of the matheuristic is able to optimally solve 76 out of the 114 instances for which the optimal solution is known. For the remaining 38, the solutions obtained are very close to the optimal ones. In addition, the p-splitting procedure allows us to improve them even more, obtaining solutions in many cases better than the optimal solutions without splits in very short computing times. This confirms that considering intermediate vertices when solving the problem can lead to better solutions.

So far we have presented the results obtained with the B&C in the K-GRP(0) instances and compared these to results provided by the matheuristic. To explore the largest instance size that the B&C was capable of optimally solving (or finding

TABLE 6 Comparison of results with single-purpose versus multi-purpose drones.

			Cost				
			Single purpose				
	K	♯ inst	Nodes	Edges	Nodes + Edges	Multipurpose	Imp (%)
General	2	5	3331.2	4377.4	7708.6	6146.1	20.35
	3	5	3916.4	4608.9	8525.3	6764.3	20.79
	4	5	4372.1	4901.2	9273.3	7362.2	20.70
	5	5	4711.8	5424.9	10136.7	8269.4	18.28
	6	5	4907.5	6092.0	10999.5	9203.1	15.71
Unit	2	5	3066.0	4359.3	7425.3	5830.9	21.63
	3	5	3239.5	4667.9	7907.4	6306.9	20.24
	4	5	3819.6	5022.1	8841.7	7009.1	20.47
	5	5	3819.6	5512.5	9332.0	7954.6	14.41
	6	5	4225.1	6130.6	10355.7	8777.8	14.87



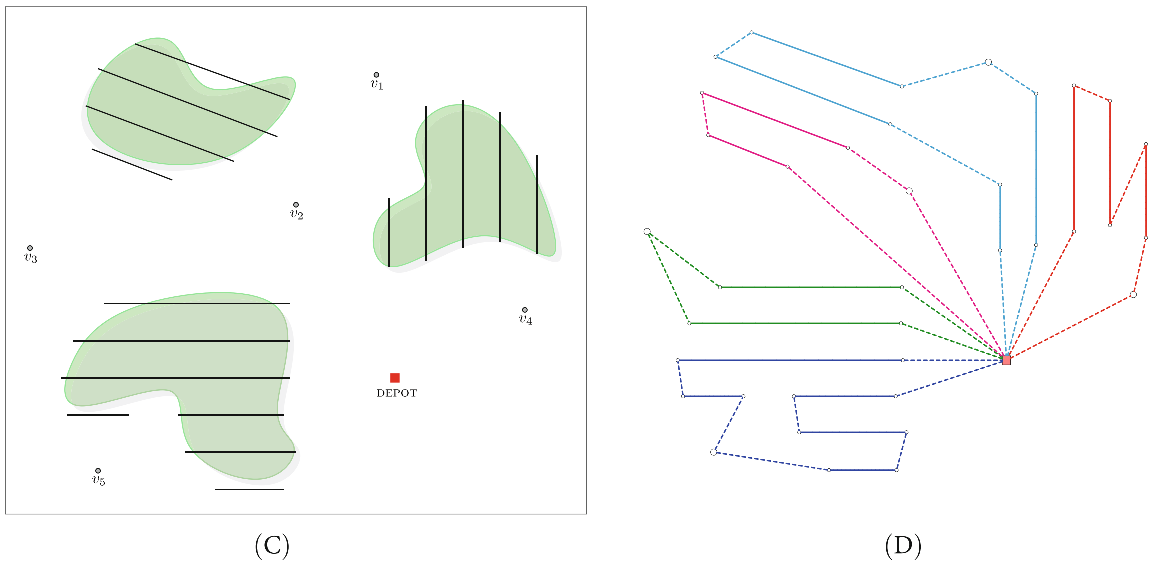


FIGURE 7 Solutions for two MP K-DGRP instances with 3 continuous areas and 5 locations. (A) A set L^1 of lines covering the required areas, (B) solution of the instance with $E_R = L^1$, (A) A set L^1 of lines covering the required areas, (B) solution of the instance with $E_R = L^1$, (C) A set L^2 of lines covering the required areas, (D) solution of the instance with $E_R = L^2$.

good upper and lower bounds), we have also run it on all K-GRP(1) instances. As we suspected, the exact algorithm was not able to optimally solve most instances with more than 60 vertices. In particular, in Type II instances, it was only able to find 3 optima out of 50 Version 1 instances and 10 out of 50 Version 2 instances. It was also unable to find a feasible solution for most of those instances. Then, we report only the results obtained on the Type I instances. They are shown in Table 5.

Table 5 shows the same figures as in Tables 2 and 3, except the column "Imp (%)" that reports the average percentage improvement of the solutions obtained in the K-GRP(1) instances with respect to those obtained for their corresponding K-GRP(0) instances. The results provided by the B&C in the instances with $|V| \leq 60$ are very good, since it is capable of solving 22 instances with general demands and 19 with unitary demands out of 25 of each type and the final average percentage gaps in the unsolved instances are small (less than 1.8% and 5.6% for instances with general and unitary demands, respectively). The results with more than 60 vertices are quite good for the instances with up to 4 vehicles, but the gaps for instances with 5 and 6 vehicles can be up to 11.5%. This shows the difficulty of these large instances, and that the difficulty increases with the number of vehicles. We also observe that the improvement found when intermediate vertices are added at the edges is not great, particularly in the instances with fewer vehicles. The reason for this behavior may be the specific spatial distribution of the required lines in the instances considered here, which does not favor that several drones share its service.

Finally, to get an idea of how much the cost of the solutions is improved by using multi-purpose drones versus using single-purpose drones (to service nodes or edges), we have applied our B&C on all the K-GRP(0) instances of Type I with $|V| \leq 60$, first considering only the required vertices of the instances and then only the required edges. We assume that two fleets of drones (K drones with delivery capability and K with inspection capability) are available to perform each type of task (it may be the same K drones that are reconfigured for missions of different type). The results obtained in these 50 instances are summarized in Table 6. Each row of this table reports the average data for the 5 solved instances. Columns 3 and 4 provide the total cost (on average) of the solutions if we consider drones that only service the required nodes (delivery only) and drones that only traverse all the required edges (sensing only), respectively. The sum of both of these costs for single purpose drones is shown in Column 5. The average costs of the solutions obtained with multi-purpose are shown in Column 6, and the last column gives the average percentage improvement that these costs represent with respect to the sum of the costs provided by single-purpose drones. It can be seen that the use of a fleet of multi-purpose drones improves the total cost between 14% and 21% (on average) compared to the use of two fleets of single-purpose drones.

5.3 | Extensions

In this work, we have assumed that the areas to be mapped were already described by means of a given set of parallel lines. However, there may be multiple possible orientations of these lines, which can greatly influence the quality of the resulting solutions. For example, consider the case illustrated in Figure 7. Figure 7A,B is the solutions obtained with the matheuristic for

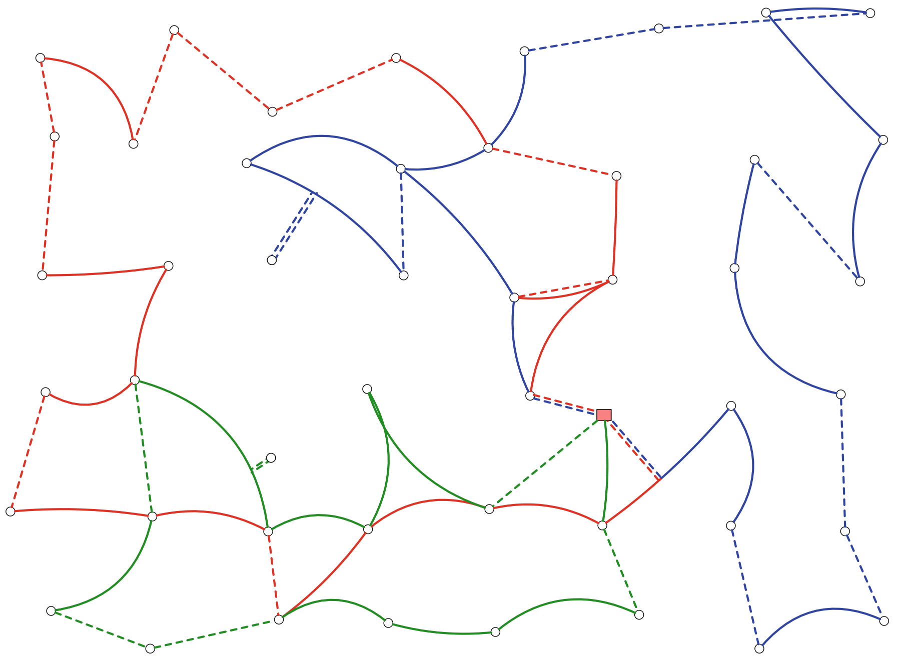


FIGURE 8 Solution of an instance with 8 required vertices and 6 connected components induced by 37 required edges corresponding to a real network.

the MP K-DGRP instances represented in Figure 7A,B, which correspond to the same areas, but with different orientations of the sets of lines that cover them. The value of the second solution is 6.2% shorter than the first one. Studying the optimal way of covering the areas is therefore an interesting problem that we would like to investigate in the future.

The solution methods described in this article, the branch and cut and the matheuristic, can be adapted to deal with general networks associated with required vertices and connected components induced by the required edges (rather than sets of parallel required lines). An example of such a network and a solution to it with three multi-purpose drones can be seen in Figure 8. This includes a connected component with 20 required edges serviced by the three drones. We also plan to extend the current work by developing versions of both methods that exploit the particularities of these general networks.

6 | CONCLUSIONS AND FUTURE WORK

In this article, we have addressed the multi-purpose K-drones general routing problem, where a fleet of multi-purpose drones are used to jointly provide sensing over a region or network along with deliveries to discrete nodes. The continuous areas for sensing coverage can be modeled as a set of parallel lines so that each area is completely serviced if all its lines are traversed. For this problem, we have proposed a formulation of its discretized version and presented a matheuristic algorithm and a branch-and-cut procedure for its solution. The results obtained with the branch-and-cut show that the formulation and the valid inequalities are useful for optimally solving small and medium-size instances and provide good lower bounds for larger instances that allow us to measure the quality of the feasible solutions provided by the matheuristic. This matheuristic algorithm is capable of finding very good solutions in short computing times.

Future research could consider a variety of practical extensions for multi-purpose drones. One area that applies for some delivery settings is to prioritize either deliveries or sensing. Thus one could enforce a requirement to complete deliveries first (before sensing activities) or vice versa. One might also enforce a time constraint on the last delivery, as can be important for medical supplies or perishable products. Another extension could be to allow drone recharging, so the drone need not return to the base, but could be recharged (or reloaded for more deliveries) in the field. A related area of future research could model the drone battery consumption as depending on the different activities (flying loaded, flying empty, and sensing). Research could also explore the optimal orientation of lines to cover a region, or explore other drone flight paths for covering a region. From an algorithmic perspective, future research can address other ways of identifying intermediate points where drones can enter and exit lines, and other heuristic solution approaches for very large problems.

ACKNOWLEDGMENTS

The work by Ángel Corberán, Isaac Plana, José M. Sanchis, and Paula Segura was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades (MICIU) and Fondo Social Europeo (FSE) through project PGC2018-099428-B-I00 and by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe" through project PID2021-122344NB-I00.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Arc Routing Problems: Data Instances at https://www.uv.es/plani/instancias.htm.

ORCID

James 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
Paula Segura https://orcid.org/0000-0002-0395-9263

REFERENCES

[1] Aerolab (2022). *XAG P40 Multipurpose Spray Drone (20L)*, Aerolab, New Zealand. https://www.aerolab.co.nz/collections/agriculturaldrones/products/xagp40.
[2] F. Barahona and M. Grötschel, *On the cycle polytope of a binary matroid*, J. Combinat. Theory B **40** (1986), 40–62.
[3] J. E. Beasley, *Route first–cluster second methods for vehicle routing*, Omega **11** (1983), 403–408.
[4] J. F. Campbell, Á. Corberán, I. Plana, and J. M. Sanchis, *Drone arc routing problems*, Networks **72** (2018), 543–559.
[5] J. F. Campbell, Á. Corberán, I. Plana, J. M. Sanchis, and P. Segura, *Solving the length constrained K–drones rural postman problem*, Eur. J. Oper. Res. **292** (2021), 60–72.

[6] D. G. Clark, J. D. Ford, and T. Tabish, *What role can unmanned aerial vehicles play in emergency response in the arctic: A case study from Canada*, PLoS One **13** (2018), no. 12, e0205299.
[7] Á. Corberán, I. Plana, A. M. Rodríguez-Chía, and J. M. Sanchis, *A branch–and–cut algorithm for the maximum benefit Chinese postman problem*, Math. Program. **141** (2013), 21–48.
[8] 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.
[9] N. V. Cuong, Y. W. P. Hong, and J. P. Sheu, *UAV trajectory optimization for joint relay communication and image surveillance*, IEEE Trans. Wirel. Commun. **21** (2022), 10177–10192.
[10] DJI Agriculture (2022). What does agricultural drone spraying of 66.7 million hectares mean to the planet?, https://ag.dji.com/newsroom/dji-ag-news-en-greener.
[11] A. Duarte, N. Mladenovic, J. Sánchez-Oro, and R. Todosijevic, "*Variable neighborhood descent*," *Handbook of Heuristics*, R. Martí, P. Pardalos, and M. Resende (eds.), Springer International Publishing AG, Cham, 2018.
[12] B. Galle, S. Arellano, N. Bobrowski, V. Conde, T. P. Fischer, G. Gerdes, A. Gutmann, T. Hoffmann, I. Itikarai, T. Krejci, E. J. Liu, K. Mulina, S. Nowicki, T. Richardson, J. Rüdiger, K. Wood, and J. Xu, *A multi-purpose, multi-rotor drone system for long-range and high-altitude volcanic gas plume measurements*, Atmos. Meas. Tech. **14** (2021), 4255–4277.
[13] G. Ghiani and G. Laporte, *A branch–and–cut algorithm for the Undirected Rural Postman Problem*, Math. Program. **87** (2000), 467–481.
[14] Unmanned Systems Technology (2022). *Sonda X8 UAS, Foldable Multi-Mission Octocopter*, https://www.unmannedsystemstechnology.com/company/hd-air-studio/sonda-x8-uas/.
[15] C. Hierholzer, *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohneUnterbrechung zu umfahren*, Math. Ann. **6** (1873), 30–32.
[16] M. Khosravi, S. Enayati, H. Saeedi, and H. Pishro-Nik, *Multi-Purpose Drones for Coverage and Transport Applications*, IEEE Trans. Wirel. Commun. **20** (2021), 3974–3987.
[17] N. Kumar, M. Ghosh, and C. Singhal, "*UAV Network for Surveillance of Inaccessible Regions with Zero Blind Spots*," *IEEE INFOCOM 2020-IEEE Conf. Comput. Commun. Workshops*, *Canada*, Toronto, 2020, pp. 1213–1218.
[18] A. N. Letchford, G. Reinelt, and D. O. Theis, *Odd minimum cut–sets and b–matchings revisited*, SIAM J. Discr. Math. **22** (2008), 1480–1487.
[19] E. McCormack and J. Stimberis, *Small unmanned aircraft evaluated for avalanche control*, Transp. Res. Rec. **2169** (2010), 168–173.
[20] N. Mladenovic and P. Hansen, *Variable neighborhood search*, Comput. Oper. Res. **24** (1997), 1097–1100.
[21] W. K. New and C. Y. Leow, "*Unmanned Aerial Vehicle (UAV) in Future Communication System*," in *26th IEEE Asia-Pacific Conf. Commun. (APCC)*, IEEE, Kuala Lumpur, 2021, pp. 217–222.
[22] C. S. Orloff, *A Fundamental Problem in Vehicle Routing*, Networks **4** (1974), 35–64.
[23] M. W. Padberg and M. R. Rao, *Odd minimum cut–sets and b–matchings*, Math. Oper. Res. **7** (1982), 67–80.
[24] C. Prins, N. Labadi, and M. Reghioui, *Tour splitting algorithms for vehicle routing problems*, Int. J. Prod. Res. **47** (2009), no. 2, 507–535.
[25] C. Prins, P. Lacomme, and C. Prodhon, *Order–first split–second methods for vehicle routing problems: A review*, Trans. Res. C: Emerg. Technol. **40** (2014), 179–200.
[26] J. Puerto and C. Valverde, *Routing for unmanned aerial vehicles: Touring dimensional sets*, Eur. J. Oper. Res. **298** (2022), 118–136.
[27] C. Seguin, G. Blaquiére, A. Loundou, P. Michelet, and T. Markarian, *Unmanned aerial vehicles (drones) to prevent drowning*, Resuscitation **127** (2018), 63–67.
[28] M. Silvagni, A. Tonoli, E. Zenerino, and M. Chiaberge, *Multipurpose UAV for search and rescue operations in mountain avalanche events*, Geomat. Nat. Haz. Risk **8** (2017), 18–33.
[29] Swoop Aero, Strengthening health supply chains in Malawi, https://swoop.aero/solutions/malawi/.
[30] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, *Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction*, Expert Syst. Appl. **55** (2016), 441–451.
[31] G. Ulusoy, *The fleet size and mix problem for capacitated arc routing*, Eur. J. Oper. Res. **22** (1985), 329–337.
[32] UNICEF Malawi (2019). *Call for Expressions of Interest (EOI): Research, Evaluation, and Market Analysis Services Related to Multi–Purpose Unmanned Aerial Systems (UAS)*. https://www.ungm.org/Public/Notice/91388.
[33] E. Chigwedere (2021). *Summative Evaluation of the Impact of Using Drones on Population Health and Other Outcomes*, UNICEF. https://www.unicef.org/malawi/reports/summative-evaluation-impact-using-drones-population-health-other-outcomes.
[34] J. I. Vasquez-Gomez, J. C. Herrera-Lozada, and M. Olguin-Carbajal, "*Coverage path planning for surveying disjoint areas*," *Proc. Int. Conf. Unmanned Aircraft Systems (ICUAS)*, IEEE, Dallas, TX, USA, June 2018, 2018, pp. 899–904.
[35] J. I. Vasquez-Gomez, M. Marciano-Melchor, L. Valentin, and J. C. Herrera-Lozada, *Coverage Path Planning for 2D Convex Regions*, J. Intell. Robot. Syst. **97** (2020), 81–94.
[36] Y. Wang, T. Kirubarajan, R. Tharmarasa, R. J. Zargani, and N. Kashyap, *Multiperiod coverage path planning and scheduling for airborne surveillance*, IEEE Trans. Aerosp. Electron. Syst. **54** (2018), 2257–2273.
[37] J. Xie, L. R. Garcia Carrillo, and L. Jin, *Path planning for UAV to cover multiple separated convex polygonal regions*, IEEE Access **8** (2020), 51770–51785.
[38] C. H. Yang, M. H. Tsai, S. C. Kang, and C. Y. Hung, *UAV path planning method for digital terrain model reconstruction – a debris fan example*, Autom. Constr. **93** (2018), 214–230.
[39] https://medium.com/frontier-technologies-hub/multi-purpose-drone-operation-in-malawi-b4cf671b14d1.
[40] https://www.fastcompany.com/90645533/how-drones-are-aiding-in-international-development.
[41] https://www.aerosociety.com/news/multirole-lifesavers-a-new-approach-to-humanitarian-drones/.

How to cite this article: J. Campbell, Á. Corberán, I. Plana, J. M. Sanchis, and P. Segura, *The multi-purpose K-drones general routing problem*, Networks. **82** (2023), 437–458. https://doi.org/10.1002/net.22176