Coordinating drones with mothership vehicles: The mothership and drone routing problem with Graphs

Lavinia Amorosi[a,*], Justo Puerto[b,*], Carlos Valverde[c,*]

[a]*Department of Statistics, Sapienza, University of Rome, Rome, 00185, Italy*
[b]*Department of Statistics and Operations Research, University of Seville, Seville, 41012, Spain*
[c]*Department of Statistics and Operations Research, University of Seville, Seville, 41012, Spain*

Abstract

This paper addresses the optimization of routing problems with drones. It analyzes the coordination of one mothership with one drone to obtain optimal routes that have to visit some target objects modeled as general graphs. The goal is to minimize the overall weighted distance traveled by both vehicles while satisfying the requirements in terms of percentages of visits to targets. We discuss different approaches depending on the assumption made on the route followed by the mothership: i) the mothership can move on a continuous framework (the Euclidean plane), ii) on a connected piecewise linear polygonal chain or iii) on a general graph. In all cases, we develop exact formulations resorting to mixed integer second order cone programs that are compared on a testbed of instances to asses their performance. The high complexity of the exact methods makes it difficult to find optimal solutions in short computing time. For that reason, besides the exact formulations we also provide a tailored matheuristic algorithm that allows one to obtain high quality solutions in reasonable time. Computational experiments show the usefulness of our methods in different scenarios.

Keywords: Arc Routing Problems, Networks, Drones, Conic Programming

1. Introduction

In recent years the progress in the field of automation has led to the increasingly widespread use of drone technology in many sectors (see [20] and [9] for a survey in civil applications). Depending on the application, these devices are used to support or replace humans in carrying out operations, and also in the cases of lack of infrastructures (see [21] also for future applications and research directions). We can find several examples of this phenomenon in the telecommunication field, where drones can be used to provide connectivity in rural areas, without antennas, or in areas affected by natural disasters which have compromised existing infrastructure (see for example [2], [7], [13] and [6]). In goods delivery activities, especially in the last mile, drones represent a valid tool to support or replace the tasks of drivers by speeding up the service and relieving traffic from big cities or cities with particular configurations where standard vehicles cannot proceed (see for example [10] and [1]). This technology allows to provide a faster and safer response even in emergency contexts, for example for the delivery of medicines or blood bags, [27]. Other uses are also for achieving safer and faster activities of inspection and monitoring, both of networks (such as electricity, gas, telecommunications, railways, roads, etc.) and areas or their portions, depending on the application context. Indeed drones can reach sections of the network that have suffered damage quickly to verify the actual conditions (for example road networks after a storm, electrical or telecommunication networks that have suffered a breakdown, etc.), or allow, for example, to check the state and progress of a fire or an oil spill at sea. The use of this technology in all these different contexts is made advantageous by the fact that compared to traditional means of transportation (trucks, ships, helicopters) Unmanned Aerial Vehicles (UAVs) have a lower cost per mile, produce less CO2 emissions and can arrive in places that cannot be reached with traditional means. On the other hand, their main limitation is the limited flight time which does not make them usable in full autonomy in a number of contexts. For this reason, for some applications, hybrid systems that involve the combined use of drones with other means of transport may represent a more efficient alternative. In this system configurations it

*Equally contributing authors

Email addresses: `lavinia.amorosi@uniroma1.it` (Lavinia Amorosi), `puerto@us.es` (Justo Puerto), `cvalverde@us.es` (Carlos Valverde)

is necessary to coordinate and synchronize the operations of drones and other means of transport, taking into account the constraints of limited autonomy of the drones and the movement of the other means of transport involved. The different configurations from which these hybrid systems can be characterized have given rise in the literature to different combinatorial optimization problems. Most of the works in the literature have focused on approaches that involve a discretization of the movement space of all the means of transport involved in the system. This provides the advantage of being able to mathematically model the problem more easily and obtain linear or linearizable formulations, but on the other hand it does not allow to fully exploit the freedom of movement of the drone which, unlike other means of transport, constrained to the road network, can move from a point to any other in the continuous space. This paper studies the problem of coordinating a system composed of a mothership (the base vehicle) which supports the operations of one drone which have to visit a set of targets represented by graphs, with the goal of minimizing the total distance travelled by both vehicles. This system configuration can model, for example, monitoring and inspection activities like the ones previously mentioned. Differently from previous works in the literature, we assume that the base vehicle and the drone can move freely on the continuous space and we present new Mixed Integer Non-Linear Programming (MINLP) formulations for this problem and a heuristic algorithm derived from the formulation to deal with larger instances. Moreover, we considered other different situations for the mothership movements: the case in which it is constrained to move on a closed polygonal and the case of a general undirected network. Also for these cases we proposed alternative MINLP formulations.
The work is structured as follows: Section 2 provides a detailed description of the problem under consideration. Section 3 reports the state of the art on routing problems with drones, mainly focusing on hybrid systems involving different means of transport. Section 4 presents alternative MINLP formulations proposed to model the different variants of the problem. Section 5 provides upper and lower bounds on the big-M constants introduced in the proposed formulations. Section 6 presents the details of the matheuristic algorithm designed to handle large instances. In Section 7 we report the results obtained testing the formulations presented in Section 4 on different classes of planar graphs and the comparison with the ones provided by the matheuristic procedure in order to evaluate its effectiveness. Finally, Section 8 concludes the paper.

2. Description of the Problem

In the Mothership and Drone Routing Problem with Graphs (MDRPG), there is one mothership (the base vehicle) and one drone, and the problem consists on the coordination between the drone and the base vehicle to minimize the total distance travelled by both vehicles. In this case, for the sake of simplicity, it is assumed that there exist no obstacles to prevent drone travelling in straight line. Nevertheless, that extension is interesting to be further considered although is beyond the scope of this paper.
The mothership and the drone begin at a starting location, denoted *orig* and a set $\mathcal{G}$ of target locations modeled by graphs, that must be visited by the drone, are located in the plane. These assumptions permit to model several real situations like roads or wired networks inspection. For each stage $t \in \{1, \ldots, |\mathcal{G}|\}$, we require that the drone is launched from the current mothership location, that at stage t is a decision variable denoted by x_L^t, flies to one of the graphs g that has to be visited , traverses the required portion of g and then returns to the current position of the mothership, that most likely is different from the launching point x_L^t, and that is another decision variable denoted by x_R^t. Once all targets graphs have been visited, the mothership and drone return to a final location (depot), denoted by *dest*.
Let $g = (V_g, E_g)$ be a graph in $\mathcal{G}$ whose total length is denoted by $\mathcal{L}(g)$ and e_g that denotes the edge e of this graph g. This edge is parametrized by its endpoints B^{e_g}, C^{e_g} and its length $\|\overline{B^{e_g}C^{e_g}}\|$ is denoted by $\mathcal{L}(e_g)$. For each line segment, we assign a binary variable μ^{e_g} that indicates whether or not the drone visits the segment e_g and define entry and exit points (R^{e_g}, ρ^{e_g}) and (L^{e_g}, λ^{e_g}), respectively, that determine the portion of the edge visited by the drone.
We have considered two modes of visit to the targets graphs $g \in \mathcal{G}$:

- Visiting a percentage α^{e_g} of each edge e_g which can be modeled by using the following constraints:

$$|\lambda^{e_g} - \rho^{e_g}|\mu^{e_g} \geq \alpha^{e_g}, \quad \forall e_g \in E_g. \tag{α-E}$$

- Visiting a percentage α_g of the total length of the graph:

$$\sum_{e_g \in E_g} \mu^{e_g}|\lambda^{e_g} - \rho^{e_g}|\mathcal{L}(e_g) \geq \alpha^g \mathcal{L}(g), \tag{α-G}$$

where $\mathcal{L}(g)$ denotes the total length of the graph.

In both cases, we need to introduce a binary variable entry^{e_g} that determines the traveling direction on the edge e_g as well as the definition of the parameter values $\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g}$ of the access and exit points to that segment. Then, for each edge e_g, the absolute value constraint (α-E) can be represented by:

$$\mu^{e_g}|\rho^{e_g}-\lambda^{e_g}| \geq \alpha^{e_g} \iff \left\{\begin{array}{rcl} \rho^{e_g}-\lambda^{e_g} & = & \nu_{\max}^{e_g}-\nu_{\min}^{e_g} \\ \nu_{\max}^{e_g} & \leq & 1-\text{entry}^{e_g} \\ \nu_{\min}^{e_g} & \leq & \text{entry}^{e_g}, \\ \mu^{e_g}(\nu_{\max}^{e_g}+\nu_{\min}^{e_g}) & \geq & \alpha^{e_g} \end{array}\right. \tag{α-E}$$

The linearization of (α-G) is similar to (α-E) by changing the last inequality in (α-E) for

$$\sum_{e_g\in E_g} \mu^{e_g}(\nu_{\max}^{e_g}+\nu_{\min}^{e_g})\mathcal{L}(e_g) \geq \alpha_g \mathcal{L}(g). \tag{α-G}$$

In our model wlog, we assume that the mothership and drone do not need to arrive at a rendezvous location at the same time: the faster arriving vehicle may wait for the other at the rendezvous location. In addition, we also assume that vehicles move at constant speeds, although this hypothesis could be relaxed. The mothership travels at v_M speed whereas the drone has a speed of $v_D > v_M$. The mothership and the drone must travel together from *orig* to the first launching point. Similarly, after the drone visits the last target location, the mothership and the drone must meet at the final rendezvous location before traveling together back to *dest*. The first launching location and final rendezvous location are allowed to be *orig* and *dest*, respectively, but it is not mandatory. For the ease of presentation, in this paper we will assume that *orig* and *dest* are the same location. However, all results extend easily to the case that *orig* and *dest* are different locations.
The goal is to find a minimum time path that begins at *orig*, ends at *dest*, and where every $g \in \mathcal{G}$ is visited by the drone.
Depending on the assumptions made on the movements of the mothership vehicle this problem gives rise to two different versions: a) the mothership vehicle can move freely on the continuous space (all terrain ground vehicle, boat on the water or aircraft vehicle); and b) the mothership vehicle must move on a road network (that is, it is a normal truck or van). In the former case, that we will call All terrain Mothership-Drone Routing Problem with Graphs (AMDRPG), each launch and rendezvous location may be chosen from a continuous space (the Euclidean 2-or-3 dimension space). In the latter case, that we will call Network Mothership-Drone Routing Problem with Graphs (NMDRPG) from now on, each launch and rendezvouz location must be chosen on a given graph embedded in the considered space. For the sake of presentation and length of the paper, we will focus in this paper, mainly, on the first model AMDRPG. The second model, namely NMDRPG, is addressed using similar techniques but providing slightly less details.

3. State of the art

In this section we focus mainly on the previous works in literature related to systems where UAVs are assisted by a vehicle in order to serve a set of targets. In these configurations the vehicle represents a launch and a recharging station for the UAVs and the main problem consists in coordinating the operations performed by one or multiple drones and the mothership. Most of the previous works in literature on this subject are focused on node routing problems (NRPs), where the vehicle moves on a road network and the drone is used to visit target points outside the road network. Many of them are related to applications in the delivery sector where the set of targets to be visited is represented by a set of customers. For example, [15] studies a delivery system consisting in one drone and one truck. The UAV visits one customer for each trip and the truck can wait at the launching node for the drone to come back or move to a different rendezvous node. In [5] the authors study a continuous approximation on the Horse Fly Problem, where the truck is used as a mobile depot for the drone. In [4] the authors evaluate the economic impact of truck-and-drone hybrid models for deliveries by means of a continuous approximation model, considering different model parameters and customer densities. [17] focuses on a delivery system where one truck supports the operations of multiple drones. The authors first clusterize the customer demand by adopting a K-means algorithm to find truck stops that represent hubs for drone

deliveries. Then, they determine a TSP of the truck among centroids of these clusters, by means of a genetic algorithm, assuming that drones are not constrained by flight range. In [16] the authors consider a similar delivery system where at each stop site the truck waits until all drones come back before moving to the next site. The goal is the minimization of the latency in a customer-oriented distribution system. Moreover, the authors compare the benefits of using drones for a single trip versus multiple trips. In [22] the authors formalize the k-Multi-visit Drone Routing Problem (k-MVDRP) considering a tandem between a truck and a fleet of k drones. The authors assumed that each drone can deliver one or more packages to customers in a single mission. Each drone may return to the truck to swap/recharge batteries, pick up a new set of packages, and launch again to customer locations. Moreover, the authors decouple the set of launch locations from the set of customer locations. The article presents a mathematical formulation including a drone energy drain function that takes into account each package weight, but the problem is then solved by means of a heuristic algorithm. [1] presents a multi-objective mixed integer linear programming model for the management of a hybrid delivery system consisting in one base vehicle and a fleet of drones. As in [22], the set of launch locations is assumed to be different from the set of customer locations. The problem consists in determining the tour of the base vehicle and the assignments of the customers to the UAVs simultaneously optimizing the distance travelled by the vehicle, the one travelled by the drones and the maximum completion time. The model is solved on two realistic urban scenarios providing a partial exploration of the Pareto frontier of the problem by means of the weighted sum method.
Other examples of similar configurations in which a base vehicle supports the operations of one or multiple drones can be found also in other sectors. [25] studies, for example, the city-scale video monitoring of a set of points of interest performed by a fleet of UAVs whose operations are supported by buses and the problem is modeled as a mixed integer linear programming model based on a multi-period directed graph. However, in all the works mentioned so far, the combined operations of vehicles and drones examine routing for a set of locations and these configurations exploit only part of the advantages of the adoption of drones. Indeed, UAVs can move between any two points in the space not following the road network. Thus, they can be adopted also for other kind of services in which the targets are represented by edges or part of them. This leads to another class of models, that is arc routing problems (ARPs).
With respected to NRPs, in the case of ARPs with drones there are relatively few papers. [19] and [18] study a coordinated road network search problem with multiple UAVs and they formulated it as a Multi-choice Multi-dimensional Knapsack Problem minimizing the flight time. This is a modified Chinese Postman Problem taking into account the UAVs energy capacity constraints. The authors solved the problem by means of a greedy insertion heuristic that models drone travel distance between the road components as a Dubins path. [11] faces the area coverage problem in sparse environments with multiple UAVs. Also in this case the UAVs motion is modeled using Dubins paths and it is assumed that they are equipped with a coverage sensor of a given radius. The edge covering problem is solved by discretizing the network in orbits and then solving a TSP among this set of orbits. The authors compares the results obtained with the ones obtained with the method by [18], for simulated urban, suburban and rural scenarios, showing that for suburban densities their method outperforms both with one or multiple UAVs. In [8] the authors deal with the problem of dynamically allocate a finite set of UAVs to links in a network that need monitoring, over multiple time periods. The need of drone monitoring is based on data related to traffic conditions. They formulated the first deterministic multi-period arc-inventory routing problem and then its stochastic dynamic version solved with an approximate dynamic programming algorithm. Another work related to multiple-period real-time monitoring of road traffic, adopting UAVs, is [14]. The authors proposed a mixed integer programming model combining the capacitated arc routing problem with the inventory routing problem and designed a local branching based method for dealing with large instances.
The ARPs with one UAV, have common characteristics with problems arising in path generation for laser cutting machines or drawing plotters. In particular, the authors of [3] studied the Drone Rural Postman Problem (DRPP) showing the relation with the Intermittent Cutting Problem (ICP). They presented a solution algorithm based on the approximation of curves in the plane by polygonal chains and that iteratively increases the number of points in the polygonal chain where the UAV can enter or leave. Thus, they solved the problem as a discrete optimization problem trying to better define the line by increasing the number of points. The authors considered also the case in which the drone has limited capacity and thus it cannot serve all the lines. To deal with this latter case, the authors assumed to have a fleet of drones and the problem consisted in finding a set of routes, each of limited length.
The literature related to Drone ARPs is limited with respect to the one related to Drone NRPs. However, as regards arc routing problems involving a hybrid system consisting in one vehicle and one or multiple

drones, the number of contributions in literature is further restricted and we can just mention [24] and [12].
In [24] the authors studied the path planning problem of a system composed of a ground robot and one drone in precision agriculture and solved it by applying orienteering algorithms. [12] studied the problem of paths planning for systems consisting in a carrier vehicle and a carried one to visit a set of target points and assuming that the carrier vehicle moves in the continuous space. In particular, the authors formulated two different problems: the first one consisting in determining the takeoff/landings sequences of the carried vehicle to visit a set of points, assuming that in each mission of the carried vehicle only one target can be visited. The second one in which this constraint is removed permitting to the carried vehicle to visit more than one target in a single mission, before returning to the carrier vehicle. Heuristic approaches have been proposed to deal with both problems.
To the best of our knowledge none of the previous papers in literature deals with a drone arc routing problem combined with a node routing problem for a hybrid system consisting of one vehicle and one drone. In this paper we present new Mixed Integer Non-Linear Programming (MINLP) formulations for the problem of coordinating a system composed of a mothership (the base vehicle) which supports the operations of one drone which has to visit a set of targets represented by graphs, with the goal of minimizing the total distance travelled by both vehicles. Indeed, differently from the previous works in literature, we assume the drone can move freely on the continuous space. As regards the mothership, we studied both cases in which also the base vehicle can move freely on the continuous space or it is constrained to move on a road network, of different typologies, where launching and rendezvous points can be chosen. The main contributions of this article can be summarized as follows:

- New mathematical formalization of the coordinating problem of a base vehicle and one drone in the continuous space to visit a set of targets represented by graphs;
- Design of new alternative second order cone MINLP formulations;
- A new matheuristic algorithm to deal with large instances able to provide high quality solutions in short time;
- Extensive experimental analysis comparing the exact solution of the formulations and the matheuristic on a set of generated instances involving different typology of planar graphs.

4. Mixed Integer Non Linear Programming Formulations

In this section we present alternative MINLP formulations for the (MDRPG) depending on the nature of the mothership that will be compared computationally in later sections. We start analyzing first the situation where the mothership moves in a continuous space, namely the (AMDRPG).

4.1. All terrain Mothership-Drone Routing Problem with Graphs

In this problem, we assume that the mothership is allowed to move freely in a continuous space: $\mathbb{R}^2$ or $\mathbb{R}^3$. In addition, the lengths are given by the Euclidean distance, $\|\cdot\|_2$, between their endpoints, although this assumption can be extended to the l_p norm, $1 \leq p \leq \infty$.

4.1.1. A first formulation for AMDRPG *based on stages*

A first attempt to model this problem uses stages identified with the order in which the different elements in the problem are visited. We identify each visit to one of the target graphs with a stage of the process. Then, by using the notation below, we define the stages set associated to graphs $T := \{1, \dots, |\mathcal{G}|\}$ and those associated to the launching and rendezvous points including *orig* and *dest* $T' = \{0, \dots, |\mathcal{G}| + 1\}$. For each stage $t \in T$, the drone follows the following path starting from the mothership, visiting the required edges of g and returning to the mothership:

$$x_L^t \rightarrow R^{e_g} \rightarrow L^{e_g} \rightarrow \dots \rightarrow R^{e'_g} \rightarrow L^{e'_g} \rightarrow \dots \rightarrow R^{e''_g} \rightarrow x_R^t \rightarrow x_L^{t+1}.$$

This path calls in a natural way for the definition of binary variables that choose:

- The optimal order to visit each graph $g \in \mathcal{G}$. In other words, defining the sequence of the stages.
- The optimal order to visit the edges of each graph in its corresponding stage.

Thus, to this end one can define the following binary variables:

- $u^{e_g t} = 1$ if the drone enters by the segment e_g at the stage t.
- $z^{e_g e'_g} = 1$ if the drone goes from the segment e_g to the segment e'_g.
- $v^{e_g t} = 1$ if the drone exits the graph by the segment e_g at the stage t.

By using these binary variables, we can model the route that follows the drone:

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t} = 1, \qquad \forall t\in T \tag{1}$$

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t} = 1, \qquad \forall t\in T \tag{2}$$

$$\sum_{e_g\in E_g}\sum_{t\in T} u^{e_g t} = 1, \qquad \forall g\in\mathcal{G} \tag{3}$$

$$\sum_{e_g\in E_g}\sum_{t\in T} v^{e_g t} = 1, \qquad \forall g\in\mathcal{G} \tag{4}$$

$$\sum_{e_g\in E_g} u^{e_g t} = \sum_{e_g\in E_g} v^{e_g t}, \qquad \forall g\in\mathcal{G}, \forall t\in T \tag{5}$$

$$\sum_{e'_g\in E_g} z_g^{e'_g e_g} + \sum_{t\in T} u^{e_g t} = \mu^{e_g}, \qquad \forall e_g\in E_g : g\in\mathcal{G} \tag{6}$$

$$\sum_{e'_g\in E_g} z_g^{e_g e'_g} + \sum_{t\in T} v^{e_g t} = \mu^{e_g}, \qquad \forall e_g\in E_g : g\in\mathcal{G} \tag{7}$$

Equations (1) and (2) state that in each stage the drone visits (enter and exit, respectively) only one graph. Constraints (3) and (4) assure that each graph is visited at some stage. Constraints (6) (resp. (7)) state that the number of exterior edges plus the number of interior edges that enter (resp. exit) to the edge e_g is given by μ^{e_g}.

Elimination of subtours
To prevent the existence of subtours within each graph $g\in\mathcal{G}$ that the drone must visit, one can include, among others, either the compact formulation that uses the Miller-Tucker-Zemlin constraints (MTZ) or the subtour elimination constraints (SEC).
For the MTZ formulation, we define the continuous variables s^{e_g} that indicate the order to visit the edge e_g and establish the following constraints for each $g\in\mathcal{G}$:

$$s^{e_g} - s^{e'_g} + |E_g| z^{e_g e'_g} \leq |E_g| - 1, \qquad \forall e_g \neq e'_g \in E_g \tag{MTZ$_1$}$$

$$0 \leq s^{e_g} \leq |E_g| - 1 \qquad \forall e_g\in E_g \tag{MTZ$_2$}$$

Alternatively, we can also use the family of subtour elimination constraints:

$$\sum_{e_g, e'_g\in S} z_g^{e_g e'_g} \leq |S| - 1, \quad \forall S\subset E_g : g\in\mathcal{G} \tag{SEC}$$

Since there is an exponential number of SEC constraints, when we implement this formulation we need to perform a row generation procedure including constraints whenever they are required by a separation oracle. To find SEC inequalities, as usual, we search for disconnected components in the current solution. Among them, we choose the shortest subtour found in the solution to be added as a lazy constraint to the model.

The goal of the AMDRPG is to find a feasible solution that minimizes the total distance traveled by the drone and the mothership. To account for the different distances among the decision variables of the model we need to define the following instrumental variables:

- $d_L^{e_g t} = \|x_L^t - R^{e_g}\|$. Distance traveled by the drone from the launching point at the stage t to the first visiting point in the segment e_g.

- $d^{e_g e'_g} = \|R^{e_g} - L^{e'_g}\|$. Distance traveled by the drone from the launching point in e_g to the rendezvous point in e'_g.
- $d^{e_g} = \|R^{e_g} - L^{e_g}\|$. Distance traveled by the drone from the rendezvous point to the next launching point in e_g.
- $d_R^{e_g t} = \|L^{e_g} - x_R^t\|$. Distance traveled by the drone from the launching point in the segment e_g to the rendezvous point on the mothership at the stage t.
- $d_{LR}^t = \|x_L^t - x_R^t\|$. Distance traveled by the mothership from the launching point to the rendezvous point at the stage t.
- $d_{RL}^t = \|x_R^t - x_L^{t+1}\|$. Distance traveled by the mothership from the rendezvous point at the stage t to the launching point at the stage $t+1$.

To ensure that the time spent by the drone to visit the graph g at the stage t is less than or equal to the time that the mothership needs to move from the launching point to the rendezvous point at the stage t, we need to define the following constraint for each $g \in \mathcal{G}$ and $t \in T$:

$$\left(\sum_{e_g \in E_g} u^{e_g t} d_L^{e_g t} + \sum_{e_g, e'_g \in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g \in E_g} v^{e_g t} d_R^{e_g t} \right) / v_D \le d_{RL}^t / v_M + M(1 - \sum_{e_g \in E_g} u^{e_g t}) \tag{DCW-t}$$

Therefore, the following formulation minimizes the overall distance traveled by the mothership and drone coordinating their movements and ensuring the required coverage of the targets.

$$\min \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \sum_{t \in T} (u^{e_g t} d_L^{e_g t} + v^{e_g t} d_R^{e_g t}) + \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{g \in \mathcal{G}} \sum_{e_g, e'_g \in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{t \in T} (d_{RL}^t + d_{LR}^t) \tag{AMDRPG-ST}$$

$$\begin{aligned}
\text{s.t.} \quad & (1) - (7), \\
& (\text{MTZ}_1) - (\text{MTZ}_2) \text{ or (SEC)}, \\
& (\alpha\text{-E}) \text{ or } (\alpha\text{-G}), \\
& (\text{DCW-t}), \\
\|x_L^t - R^{e_g}\| &\le d_L^{e_g t}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
\|R^{e_g} - L^{e_g}\| &\le d^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
\|R^{e_g} - L^{e'_g}\| &\le d^{e_g e'_g}, \quad \forall e_g \neq e'_g \in E_g : g \in \mathcal{G}, \\
\|L^{e_g} - x_R^t\| &\le d_R^{e_g t}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
\|x_R^t - x_L^{t+1}\| &\le d_{RL}^t, \quad \forall t \in T, \\
\|x_L^t - x_R^t\| &\le d_{LR}^t, \quad \forall t \in T, \\
x_L^0 &= orig, \\
x_R^0 &= orig, \\
x_L^{|\mathcal{G}|+1} &= dest, \\
x_R^{|\mathcal{G}|+1} &= dest.
\end{aligned}$$

The objective function has four terms: the first one computes the distances to go to and return from the mothership to the target graph g visited at the stage t, the second one accounts for the distance traveled by the drone on the edges of the target graph g between two consecutive *jumps* between edges, the third one computes the distances traveled by the drone while it *jumps* between two consecutive edges on the target graph g and the fourth term expresses the distances made by the mothership. Constraints (1)-(7) models the route followed by the drone, (MTZ$_1$) - (MTZ$_2$) or (SEC) ensure that the displacement of the drone in each visit to a target graph is a route, (α-E) or (α-G) defines what is required in each visit to a target graph. Then, the next six constraints model Euclidean distances needed in the model. Observe that we are assuming constant velocities for the mothership v_M and the drone v_D.

To deal with the bilinear terms of the objective function, we use McCormick's envelopes to linearize these terms by adding variables $p \geq 0$ that represent these products and we introduce these constraints:

$$\begin{aligned} p &\geq mz, \\ p &\leq d - M(1-z), \end{aligned}$$

where m and M are, respectively, the lower and upper bounds of the distance variable d. These bounds will be adjusted for each bilinear term in Section 5.

4.1.2. Alternative formulations for AMDRPG *based on enforcing connectivity: MTZ-constraints and subtour elimination-constraints (SEC)*

In this family of formulations we replace the variables $u^{\cdot t}$, $v^{\cdot t}$ and constraints that model the tour using stages, namely (1)-(7), by constraints that ensure connectivity. We will distinguish two different approaches. The first one is based on the rationale of Miller-Tucker-Zemlin (MTZ) whereas the second one uses an extended formulation based on SEC.

We start presenting the formulation based on MTZ. It requires the definition of two families of binary variables that choose:

- The optimal order to visit the graphs $g \in \mathcal{G}$ in the tour of the mothership.
- The optimal order to visit the edges of each graph.

The above concepts can be modeled with the following binary variables:

- $u^{e_g} = 1$ if the drone accesses the segment e_g.
- $z^{e_g e'_g} = 1$ if the drone goes from the segment e_g to the segment e'_g.
- $v^{e_g} = 1$ if the drone exits the graph g by the segment e_g.
- $w^{gg'} = 1$ if the mothership goes from x_L^g to $x_R^{g'}$.

By using these binary variables, we can model the route that follows the drone in each particular graph $g \in \mathcal{G}$:

$$\sum_{e_g \in E_g} u^{e_g} = 1, \tag{8}$$

$$\sum_{e_g \in E_g} v^{e_g} = 1, \tag{9}$$

$$\sum_{e'_g \in E_g} z^{e'_g e_g} + u^{e_g} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G} \tag{10}$$

$$\sum_{e'_g \in E_g} z^{e_g e'_g} + v^{e_g} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G} \tag{11}$$

On the other hand, to model the tour followed by the mothership, we have to include the following new constraints:

$$\sum_{g \in \mathcal{G}} w^{g0} = 0, \tag{12}$$

$$\sum_{g' \in \mathcal{G}} w^{(n_G+1)g'} = 0, \tag{13}$$

$$\sum_{g' \in \mathcal{G} \setminus \{g\}} w^{gg'} = 1, \qquad \forall g \in \mathcal{G} \tag{14}$$

$$\sum_{g \in \mathcal{G} \setminus \{g'\}} w^{gg'} = 1, \qquad \forall g' \in \mathcal{G} \tag{15}$$

$$s_g - s_{g'} + |\mathcal{G}| w^{gg'} \leq |\mathcal{G}| - 1, \qquad \forall g \neq g' \tag{MTZ$_3$}$$

$$0 \leq s_g \leq |\mathcal{G}| - 1 \qquad \forall g \in \mathcal{G} \tag{MTZ$_4$}$$

$$s_0 = 0, \tag{MTZ$_5$}$$

$$s_{n_G+1} = n_G + 1, \tag{MTZ$_6$}$$

The reader may note that constraints (14), (15) and (MTZ_3)-(MTZ_6) are of the type MTZ for the complete graph $\hat{G} = (\hat{N}, \hat{E})$ induced by the graphs $g \in \mathcal{G}$ to be visited, where a node $\hat{v}_g \in \hat{N}$ if $g \in \mathcal{G}$. Thus, we enforce that the mothership route follows a Hamiltonian tour on the graphs to be visited by the drone. Next, we have to connect the visits of the drone to the graphs with the tour followed by the mothership. To this end, we require to introduce variables defining the launching and recovery points to account for the distances traveled and to define the inner tour of the mothership.

In a similar way, we need to define the following continuous variables that account for the distances among distinguished points:

- $d_L^{e_g} = \|x_L^g - R^{e_g}\|$. Distance traveled by the drone from the launching point to the first visiting point in the segment e_g.
- $d^{e_g e'_g} = \|R^{e_g} - L^{e'_g}\|$. Distance traveled by the drone from the launching point in e_g to the next rendezvous point e'_g.
- $d^{e_g} = \|R^{e_g} - L^{e_g}\|$. Distance traveled by the drone from the rendezvous point to the next launching point in e_g.
- $d_R^{e_g} = \|L^{e_g} - x_R^g\|$. Distance traveled by the drone from the launching point in the segment e_g to the rendezvous point on the mothership.
- $d_{LR}^g = \|x_L^g - x_R^g\|$. Distance traveled by the mothership from the launching point to the rendezvous point while the drone is visiting g.
- $d_{RL}^{gg'} = \|x_R^g - x_L^{g'}\|$. Distance traveled by the mothership from the rendezvous point for the graph g to the launching point associated to the graph g'.

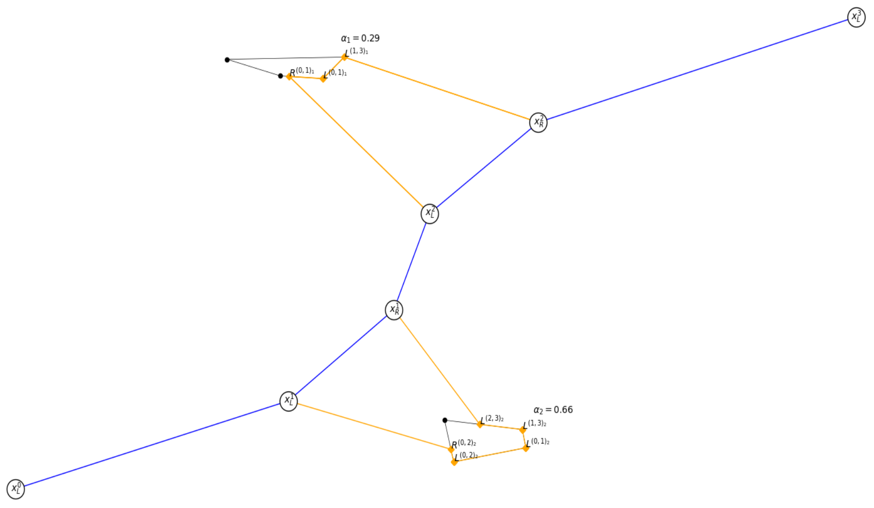

Figure 1: Example illustrating the meaning of the launching (L) and rendezvous (R) points.

Figure 1 shows an example of a configuration with two target graphs with four nodes and four edges. The mothership begins at its starting point x_L^0. Then, it moves to x_L^1 where the drone is launched to visit the graph 2. There, the drone follows a route that ensures the coverage of 66% of the entire graph and returns to the point x_R^2 from where it is launched again to visit the second graph. Once, this graph is visited the drone returns to the mothership at the rendezvous point x_R^2 and so on.
In this formulation of AMDRPG the goal is again to find a feasible solution that minimizes the total distance traveled by the drone and the mothership but without using stages. Instead the model ensures

the Hamiltonian nature of the routes using the rationale of connectivity by the MTZ or SEC family of constraints.
Again, we need to be sure that the time spent by the drone to visit the graph g is less than or equal to the time that the mothership needs to move from the launching point to the rendezvous point associated to this graph g. Hence, by using the same argument, as the one used in (DCW-t), we define for each $g \in \mathcal{G}$:

$$\left(\sum_{e_g \in E_g} u^{e_g} d_L^{e_g} + \sum_{e_g, e'_g \in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g \in E_g} v^{e_g} d_R^{e_g} \right) / v_D \leq d_{RL}^g / v_M, \quad \forall g \in \mathcal{G}. \tag{DCW-g}$$

Therefore, modifying the previous formulation (AMDRPG-ST), replacing constraints based on stages, one obtains the following alternative valid formulation.

$$\min \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} (u^{e_g} d_L^{e_g} + v^{e_g} d_R^{e_g}) + \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{g \in \mathcal{G}} \sum_{e_g, e'_g \in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{g \in \mathcal{G}} d_{LR}^g + \sum_{g, g' \in \mathcal{G}} d_{RL}^{gg'} w^{gg'} \tag{AMDRPG-MTZ}$$

$$\begin{aligned}
\text{s.t.} \quad & (8) - (11), \\
& (12) - (15), \\
& (\mathrm{MTZ}_1) - (\mathrm{MTZ}_2) \text{ or } (\mathrm{SEC}), \\
& (\mathrm{MTZ}_3) - (\mathrm{MTZ}_6), \\
& (\alpha\text{-E}) \text{ or } (\alpha\text{-G}), \\
& (\text{DCW-g}), \\
\|x_L^g - R^{e_g}\| &\leq d_L^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \\
\|R^{e_g} - L^{e_g}\| &\leq d^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \\
\|R^{e_g} - L^{e'_g}\| &\leq d^{e_g e'_g}, \quad \forall e_g \neq e'_g : g \in \mathcal{G}, \\
\|L^{e_g} - x_R^g\| &\leq d_R^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \\
\|x_R^g - x_L^{g'}\| &\leq d_{RL}^{gg'}, \quad \forall g, g' \in \mathcal{G}, \\
\|x_L^g - x_R^g\| &\leq d_{LR}^g, \quad \forall g \in \mathcal{G}, \\
x_L^0 &= orig, \\
x_R^0 &= orig, \\
x_L^{|\mathcal{G}|+1} &= dest, \\
x_R^{|\mathcal{G}|+1} &= dest.
\end{aligned}$$

The formulation above (AMDRPG-MTZ) can be slightly modified replacing constraints (MTZ_3) – (MTZ_6) by

$$\sum_{g, g' \in \mathcal{G}} w^{gg'} \leq |S| - 1, \quad \forall S \subseteq \{1, \ldots, |\mathcal{G}|\}. \tag{16}$$

In doing so we obtain another formulation for AMDRPG using SEC rather than MTZ constraints to represent the mothership tour. We will call this formulation (AMDRPG-SEC). Later, we will compare their performance in the computational results section.

4.1.3. Comparing the formulations

A straightforward analysis of the two types of formulations, namely by stages and based on connectivity, shows that (AMDRPG-MTZ) has much less variables and constraints than (AMDRPG-ST). Indeed, the latter requires definition of distance variables for each e_g and t which is a quadratic function rather than a linear one as used by the former. In addition, to enforce coordination (AMDRPG-ST) needs constraints (DCW) for each $g \in \mathcal{G},\ t \in T$ whereas (AMDRPG-MTZ) only uses (DCW) constraints for $g \in \mathcal{G}$. This is an important decrease in the dimension of the problems to be solved. This observation is confirmed in efficiency in problem solving discussed in the experimental results section.

*4.2. Network Mothership-Drone Routing Problem on a Polygonal with Graphs (*PMDRPG*)*

The most significant difference between AMDRPG and PMDRPG is that, in the latter, the mothership cannot move freely in $\mathbb{R}^2$ but it has to move on a network that models a road system where the mothership (truck) is constrained to move. An intermediate situation between moving freely and moving on a general road network is to assume that the mothership moves on a closed polygonal, namely a piecewise linear, chain. Modeling this situation is a transition step to achieve the more general case of the mothership moving on general networks. We will present the formulations for the PMDRPG following a scheme similar to that of AMDRPG. First, we present a model by stages and then another one based on connectivity constraints.

Let $\mathcal{P}$ denote a polygonal chain parametrized by its breakpoints $V^1, \dots, V^p$ ordered in the direction of travel. Observe that we are assuming that there exists a pre-specified orientation that determines the direction of the mothership's displacement. Let also γ_L^{et} and γ_R^{et} be the parameter values that define the points x_L^t and x_R^t on the edge e on $\mathcal{P}$, respectively. Figure 2 illustrates the meaning of the parameters γ_L^{et} and γ_R^{et} on a polygonal chain with 4 pieces. Then, the distance measured on the polygonal can be obtained by the following expressions for each $t \in T$:

$$d_{LR}^{ee't} = \begin{cases} |\gamma_R^{et} - \gamma_L^{et}|\mathcal{L}(e), & \text{if } e = e', \\ (1-\gamma_L^{et})\mathcal{L}(e) + \sum_{e''=e+1}^{e'-1} \mathcal{L}(e'') + \gamma_R^{e't}\mathcal{L}(e'), & \text{if } e < e', \\ \gamma_L^{et}\mathcal{L}(e) + \sum_{e''=e'+1}^{e-1} \mathcal{L}(e'') + (1-\gamma_R^{e't})\mathcal{L}(e'), & \text{if } e > e'. \end{cases} \tag{$d_{LR}^{t\mathcal{P}}$}$$

$$d_{RL}^{ee't} = \begin{cases} |\gamma_L^{et+1} - \gamma_R^{et}|\mathcal{L}(e), & \text{if } e = e', \\ (1-\gamma_R^{et})\mathcal{L}(e) + \sum_{e''=e+1}^{e'-1} \mathcal{L}(e'') + \gamma_L^{e't+1}\mathcal{L}(e'), & \text{if } e < e', \\ \gamma_R^{et}\mathcal{L}(e) + \sum_{e''=e'+1}^{e-1} \mathcal{L}(e'') + (1-\gamma_L^{e't+1})\mathcal{L}(e'), & \text{if } e > e'. \end{cases} \tag{$d_{RL}^{t\mathcal{P}}$}$$

This distance requires a binary variable $z_{LR}^{ee't}$ that determines whether $d_{LR}^{ee'}$ is defined by the first or the second expression in the formula:

$$z_{LR}^{ee't} = 1 \text{ if } \mu_L^{et} = 1 \text{ and } \mu_R^{e't} = 1,$$

where μ_L^{et} and μ_R^{et} are indicator variables that attain the value one if the launching point (resp. rendezvous point) is placed in the segment e of $\mathcal{P}$ at the stage t.

Similarly, we define $z_{RL}^{ee't}$:

$$z_{RL}^{ee't} = 1 \text{ if } \mu_R^{et} = 1 \text{ and } \mu_L^{e't+1} = 1.$$

With all this, we can model the movement of the mothership on the polygonal. For each stage $t \in T$, we have

$$x_L^t = \sum_{e=(i,i+1)\in\mathcal{P}} \mu_L^{et}\left[V^i + \gamma_L^{et}(V^{i+1} - V^i)\right] \tag{17}$$

$$x_R^t = \sum_{e=(i,i+1)\in\mathcal{P}} \mu_R^{et}\left[V^i + \gamma_R^{et}(V^{i+1} - V^i)\right] \tag{18}$$

$$z_{LR}^{ee't} = \mu_L^{et}\mu_R^{e't} \tag{19}$$

$$z_{RL}^{ee't} = \mu_R^{et}\mu_L^{e't+1} \tag{20}$$

$$\sum_e \mu_L^{et} = 1 \tag{21}$$

$$\sum_e \mu_R^{et} = 1 \tag{22}$$

$$d_{LR}^t = \sum_{e,e'} z_{LR}^{ee't} d_{LR}^{ee'}, \tag{23}$$

$$d_{RL}^t = \sum_{e,e'} z_{RL}^{ee't} d_{RL}^{ee't}, \tag{24}$$

where $d_{LR}^{ee'}$ and $d_{RL}^{ee't}$ are defined in ($d_{LR}^{t\mathcal{P}}$) and ($d_{RL}^{t\mathcal{P}}$). Constraints (17) and (18) parameterize the launching and rendezvous points on the polygonal chain. Constraints (19) and (20) model the product of binary variables. Constraints (21) and (22) state that, in each stage, the launching and rendezvous points can be only in one segment, not necessarily the same. Constraints (23) and (24) compute the total distance between the launching and the rendezvous points in each stage.

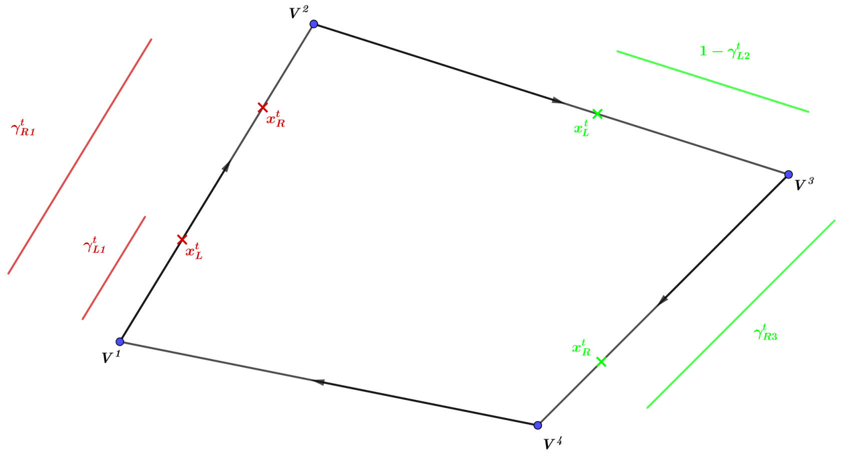

Figure 2: An example of a parameterization of a mothership route that goes from V^1 to x_L^t to launching the drone, then moves to x_R^t on the edge $\overline{V^1V^2}$ to retrieve it, then it goes to x_L^t on $\overline{V^2V^3}$ to launch it again, and travels until x_R^t on $\overline{V^3V^4}$ to retrieve the drone and returns to the starting point at V^1.

In addition, for each stage $t \in T$ we can define continuous variables λ_R^t and λ_L^t that model the absolute position of the point x_L^t in the polygonal to ensure that the movement of the mothership goes always in the allowed direction of travel and never comes back when it traverses the polygonal chain.

$$\lambda_L^t - i \geq \mu_L^{et} - (p-1)(1-\mu_L^{et}), \quad \forall e = (i, i+1) \in \mathcal{P} \tag{25}$$
$$\lambda_L^t - i \leq \mu_L^{et} - (p-1)(1-\mu_L^{et}), \quad \forall e = (i, i+1) \in \mathcal{P} \tag{26}$$
$$\lambda_R^t - i \geq \mu_R^{et} - (p-1)(1-\mu_R^{et}), \quad \forall e = (i, i+1) \in \mathcal{P} \tag{27}$$
$$\lambda_R^t - i \leq \mu_R^{et} + (p-1)(1-\mu_R^{et}), \quad \forall e = (i, i+1) \in \mathcal{P} \tag{28}$$
$$\lambda_R^t \geq \lambda_L^t \tag{29}$$
$$\lambda_L^{t+1} \geq \lambda_R^t \tag{30}$$

Inequalities (25)-(28) link μ_L^{et} and λ_L^t (resp. μ_R^{et} and λ_R^t). Finally, constraints (29) and (30) ensure that the mothership moves in the right direction on the polygonal.
The goal of the (PMDRPG) is, once again, to find a feasible solution that minimizes the total distance covered by the mothership and the drone. Thus, gathering all the above constraints one gets the following

valid formulation.

$$\min \quad \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\sum_{t\in T}(u^{e_gt}d_L^{e_gt}+v^{e_gt}d_R^{e_gt})+\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\mu^{e_g}d^{e_g}+\sum_{g\in\mathcal{G}}\sum_{e_g,e'_g\in E_g}z^{e_ge'_g}d^{e_ge'_g}+\sum_{t\in T}(d_{RL}^t+d_{LR}^t) \tag{PMDRPG}$$

$$\begin{aligned}
\text{s.t.} \quad & (1) - (7),\\
& (17) - (24),\\
& (\text{MTZ}_1) - (\text{MTZ}_2) \text{ or (SEC)},\\
& (\alpha\text{-E}) \text{ or } (\alpha\text{-G}),\\
& (\text{DCW-t}),\\
& (d_{LR}^{t\mathcal{P}}), (d_{RL}^{t\mathcal{P}}),\\
\|x_L^t - R^{e_g}\| \leq d_L^{e_gt}, \quad & \forall e_g : g\in\mathcal{G}, \forall t\in T,\\
\|R^{e_g} - L^{e_g}\| \leq d^{e_g}, \quad & \forall e_g : g\in\mathcal{G}, \forall t\in T,\\
\|R^{e_g} - L^{e'_g}\| \leq d^{e_ge'_g}, \quad & \forall e_g, e'_g : g\in\mathcal{G},\\
\|L^{e_g} - x_R^t\| \leq d_R^{e_gt}, \quad & \forall e_g : g\in\mathcal{G}, \forall t\in T,\\
x_L^0 = orig, & \\
x_R^0 = orig, & \\
x_L^{|\mathcal{G}|+1} = dest, & \\
x_R^{|\mathcal{G}|+1} = dest. &
\end{aligned}$$

4.2.1. Alternative formulation for PMDRPG *based on MTZ-constraints*

Analogously to the case of AMDRPG one can also model PMDRPG without any explicit reference to stages. Here, we will follow a very similar approach to that in Section 4.1.2 to describe the formulation for PMDRPG based on MTZ-constraints.

Let $\mathcal{P}$ denote, as before, the polygonal chain parameterized by its endpoints $V^1,\dots,V^p$ ordered in the direction of travel. In addition, let also γ_L^{eg} and γ_R^{eg} be the parameter values that define the points x_L^g and x_R^g on the line segment $\overline{V^iV^{i+1}}$ in $\mathcal{P}$, respectively. Then, the distance traveled on the polygonal can be obtained by one of the following expressions. The first one refers to distances covered by the mothership between the launching and rendezvous points of a drone operation, whereas models the distance traveled by the mothership between the launching and rendezvous points of two consecutive graphs targeted in its route.

$$d_{LR}^{ee'g} = \begin{cases} |\gamma_L^{eg} - \gamma_R^{eg}|\mathcal{L}(e), & \text{if } e = e',\\ (1-\gamma_L^{eg})\mathcal{L}(e) + \sum_{e''=e+1}^{e'-1}\mathcal{L}(e'') + \gamma_R^{e'g}\mathcal{L}(e'), & \text{if } e < e',\\ \gamma_L^{eg}\mathcal{L}(e) + \sum_{e''=e+1}^{e'-1}\mathcal{L}(e'') + (1-\gamma_R^{e'g})\mathcal{L}(e'), & \text{if } e > e'. \end{cases} \tag{$d_{LR}^{g\mathcal{P}}$}$$

$$d_{RL}^{ee'gg'} = \begin{cases} |\gamma_L^{eg} - \gamma_R^{eg'}|\mathcal{L}(e), & \text{if } e = e',\\ (1-\gamma_R^{eg})\mathcal{L}(e) + \sum_{e''=e+1}^{e'-1}\mathcal{L}(e'') + \gamma_L^{e'g'}\mathcal{L}(e'), & \text{if } e < e',\\ \gamma_R^{eg}\mathcal{L}(e) + \sum_{e''=e+1}^{e'-1}\mathcal{L}(e'') + (1-\gamma_L^{e'g'})\mathcal{L}(e'), & \text{if } e > e'. \end{cases} \tag{$d_{RL}^{g\mathcal{P}}$}$$

This distance requires binary variables $z_{LR}^{ee'g}$ and $z_{RL}^{ee'gg'}$ that determines which of the expressions in $(d_{LR}^{g\mathcal{P}})$ and $(d_{RL}^{g\mathcal{P}})$ define $d_{LR}^{ee'g}$ and $d_{RL}^{ee'gg'}$, respectively.

$$z_{LR}^{ee'g} = 1 \text{ iff } \mu_L^{eg} = 1 \text{ and } \mu_R^{e'g} = 1, \qquad z_{RL}^{ee'gg'} = 1 \text{ iff } \mu_L^{eg} = 1 \text{ and } \mu_R^{e'g'} = 1,$$

where μ_L^{eg} and μ_R^{eg} are indicator variables that attain the value one if the launching point (resp. rendezvous point) associated to the graph g is placed in the segment $\overline{V^iV^{i+1}}$ of the edge $e=(i,i+1)\in\mathcal{P}$. With all this, we can model the movement of the mothership inside the polygonal. Thus, for each $g,g'\in\mathcal{G}$ we have:

$$x_L^g = \sum_{e=(i,i+1)\in\mathcal{P}} \mu_L^{eg}\left[V^i + \gamma_L^{eg}(V^{i+1}-V^i)\right] \tag{31}$$

$$x_R^g = \sum_{e=(i,i+1)\in\mathcal{P}} \mu_R^{eg}\left[V^i + \gamma_R^{eg}(V^{i+1}-V^i)\right] \tag{32}$$

$$z_{LR}^{ee'g} = \mu_L^{eg}\mu_R^{e'g} \tag{33}$$

$$z_{RL}^{ee'gg'} = \mu_R^{eg}\mu_L^{e'g'} \tag{34}$$

$$\sum_e \mu_L^{eg} = 1 \tag{35}$$

$$\sum_e \mu_R^{eg} = 1 \tag{36}$$

$$d_{LR}^g = \sum_{e,e'} z_{LR}^{ee'g} d_{LR}^{ee'g}, \tag{37}$$

$$d_{RL}^{gg'} = \sum_{e,e'} z_{RL}^{ee'gg'} d_{RL}^{ee'gg'}, \tag{38}$$

where d_{LR}^g and $d_{RL}^{gg'}$ are defined in $(d_{LR}^{g\mathcal{P}})$ and $(d_{RL}^{g\mathcal{P}})$, respectively. Constraints (31) and (32) parameterize the launching and rendezvous points on the polygonal chain. Constraints (33) and (34) model the product of binary variables. Constraints (35) and (36) state that, in each stage, the launching and rendezvous points can be only in one segment, not necessarily the same. Constraints (37) and (38) compute the total distance between the launching and the rendezvous points for each pair of points. The reader may note that the above representation is similar to that given by (17)-(24) with the exception that in (31)-(38) constraints are indexed in the set of graphs $g \in \mathcal{G}$ instead that on the stages. This allows a remarkable reduction in the number of distance variables and constraints as already discussed in Section 4.1.3.
The goal of the (PMDRPG) is to find a feasible solution that minimizes the total distance covered by the drone and the mothership, i.e.,

$$\min \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}(u^{e_g}d_L^{e_g} + v^{e_g}d_R^{e_g}) + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\mu^{e_g}d^{e_g} + \sum_{g\in\mathcal{G}}\sum_{e_g,e'_g\in E_g} z^{e_g e'_g}d^{e_g e'_g} + \sum_{g\in\mathcal{G}} d_{LR}^g + \sum_{g,g'\in\mathcal{G}} d_{RL}^{gg'}w^{gg'} \tag{PMDRPG-MTZ}$$

$$\begin{aligned}
\text{s.t.} \quad & (8)-(11),\\
& (12)-(15),\\
& (\text{MTZ}_1)-(\text{MTZ}_2) \text{ or } (\text{SEC}),\\
& (\text{MTZ}_3)-(\text{MTZ}_6) \text{ or } (\text{SEC}),\\
& (31)-(38),\\
& (\alpha\text{-E}) \text{ or } (\alpha\text{-G}),\\
& (\text{DCW-g}),\\
& (d_{LR}^{g\mathcal{P}}), (d_{RL}^{g\mathcal{P}}),\\
\|x_L^g - R^{e_g}\| \leq d_L^{e_g}, & \quad \forall e_g : g\in\mathcal{G},\\
\|R^{e_g} - L^{e_g}\| \leq d^{e_g}, & \quad \forall e_g : g\in\mathcal{G},\\
\|R^{e_g} - L^{e'_g}\| \leq d^{e_g e'_g}, & \quad \forall e_g : g\in\mathcal{G},\\
\|L^{e_g} - x_R^i\| \leq d_R^{e_g}, & \quad \forall e_g : g\in\mathcal{G},\\
x_L^0 = orig, &\\
x_R^0 = orig, &\\
x_L^{|\mathcal{G}|+1} = dest, &\\
x_R^{|\mathcal{G}|+1} = dest. &
\end{aligned}$$

The reader may note that the above formulation actually encloses two different ones depending whether it includes constraints (MTZ$_1$) - (MTZ$_2$) or (SEC). However, for the sake of presentation we have preferred to present them in this compact form to reduce the length of the paper.

*4.3. Network Mothership-Drone Routing Problem with Graphs (*NMDRPG*)*

This new model is a refinement of the last one as here the mothership has to move on a general undirected network rather than on a polygonal chain. Even though the previous one, namely PMDRPG, can be seen as a particular case of NMDRPG, for the sake of presentation, we have preferred to include it first to ease the understanding of the more advanced NMDRPG model.

As before, we start with an approach that models the problems by stages. Later, we shall extend the analysis using the rationale of connectivity, as already done in the the previous models AMDRPG and PMDRPG.

Let $\mathcal{N} = (V, E)$ be the network that models the space of movement for the mothership. For each stage $t \in T := \{1, \ldots, |\mathcal{G}|\}$, we assume that the mothership starts and ends in the edge $e = (i, j)$, $e' = (i', j') \in E$, respectively. Then, it follows a route from the launching to the rendezvous point and vice versa, determined by a sequence of intermediate points as shown in Figure 3:

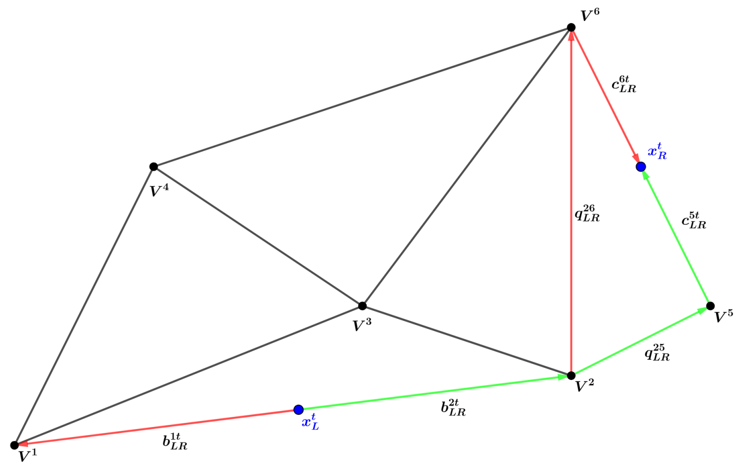

Figure 3: An example of a parameterization of a mothership route that launches the drone in the point x^t_L on the edge $\overline{V^1V^5}$, then traverses the edge $\overline{V^2V^5}$ and finally moves to x^t_R on the edge $\overline{V^5V^4}$ to retrieve it. The arrows show possible directions of traveling: the green ones are the selected to go from x^t_L to x^t_R.

Therefore, the distance traveled by the mothership, between two consecutive launching and rendezvous points in two edges, not necessarily distinct, of the graph can be represented as:

$$d^{ee't}_{LR} = \begin{cases} |\gamma^{et}_L - \gamma^{et}_R|\mathcal{L}(e), & \text{if } e = e', \\ \left[b^{it}_{LR}\gamma^{et}_L + b^{jt}_{LR}(1-\gamma^{et}_L)\right]\mathcal{L}(e) + \sum_{e''\in\mathcal{N}} q^{e''t}_{LR}\mathcal{L}(e'') + \left[c^{i't}_{LR}\gamma^{e't}_R + c^{j't}_{LR}(1-\gamma^{e't}_R)\right]\mathcal{L}(e'), & \text{if } e \neq e', \end{cases} \tag{$d^{t\mathcal{N}}_{LR}$}$$

where b^{it}_{LR} (resp. c^{it}_{LR}) are binary variables that determine from which of the end-points of e (respectively e') one has to account for the distance and q^{et}_{LR} is an integer variable that counts how many times the mothership fully traverses the edge e. Furthermore, we need to define another binary variable $z^{ee't}_{LR}$ that models the correct definition of the distance in $(d^{t\mathcal{N}}_{LR})$. With the above definition one can account for the movement of the mothership at each stage $t \in T$ from a launching to a rendezvous point:

$$x_L^t = \sum_{e=(i,j)\in E} \mu_L^{et}\left[V^i + \gamma_L^{et}(V^j - V^i)\right] \tag{39}$$

$$x_R^t = \sum_{e=(i,j)\in E} \mu_R^{et}\left[V^i + \gamma_R^{et}(V^j - V^i)\right] \tag{40}$$

$$z_{LR}^{ee't} = \mu_{Le}^t \mu_{Re'}^t, \tag{41}$$

$$b_{LR}^{it} \leq \sum_{e\in\delta(i)} \mu_L^{et}, \qquad \forall i \in V \tag{42}$$

$$c_{LR}^{it} \leq \sum_{e\in\delta(i)} \mu_R^{et}, \qquad \forall i \in V \tag{43}$$

$$b_{LR}^{it} + b_{LR}^{jt} \geq \mu_L^{et}, \qquad \forall e = (i,j) \in E \tag{44}$$

$$c_{LR}^{it} + c_{LR}^{jt} \geq \mu_R^{et}, \qquad \forall e = (i,j) \in E \tag{45}$$

$$b_{LR}^{it} + \sum_{\{j:(i,j)\in E\}} q_{LR}^{jit} = \sum_{\{j:(i,j)\in E\}} q_{LR}^{ijt} + c_{LR}^{it}, \qquad \forall i \in V \tag{46}$$

$$\sum_{e\in E} \mu_L^{et} = 1, \tag{47}$$

$$\sum_{e\in E} \mu_R^{et} = 1, \tag{48}$$

$$d_{LR}^t = \sum_{e,e'\in E} z_{LR}^{ee't} d_{LR}^{ee't}, \tag{49}$$

Constraints (39) and (40) parameterize the launching and rendezvous points in the network $\mathcal{N}$ at stage t. Constraints (41) model the product of binary variables. Constraints (42) and (43) state that if the launching point (resp. rendezvous point) is not in the edge e, the mothership cannot go (resp. exit) to the vertex that is incident to e. Constraints (44) state that if the mothership leaves the edge e to go to e', it must exit from one of the incident vertices to e. In the same way, constraints (45) express that if the mothership leaves the edge e' to go to e, it must enter to the edge e from one of its incident vertices. Flow conservation constraints (46) ensure that the number of incoming edges to each vertex i is equal to the number of outgoing edges in the route followed by the mothership. Constraints (47) and (48) state that, in each stage, the selected points can be only in one edge. Finally, constraints (49) define the total distance between the launching and the rendezvous points at stage t.

Similarly, the distance covered by the mothership along the path on the network from the rendezvous point x_R^t to the next launching point x_L^{t+1} can be modeled using the following definition of distance:

$$d_{RL}^{ee't} = \begin{cases} |\gamma_R^{et} - \gamma_L^{et+1}|\mathcal{L}(e), & \text{if } e = e', \\ \left[b_{RL}^{it}\gamma_R^{et} + b_{RL}^{jt}(1-\gamma_R^{et})\right]\mathcal{L}(e) + \sum_{e''\in\mathcal{N}} q_{RL}^{e''t}\mathcal{L}(e'') + \left[c_{RL}^{i't}\gamma_L^{e't+1} + c_{RL}^{j't}(1-\gamma_L^{e't+1})\right]\mathcal{L}(e'), & \text{if } e \neq e'. \end{cases} \tag{$d_{RL}^{t\mathcal{N}}$}$$

In this case, the binary variable $z_{ee'}^{RLt}$ links the rendezvous point at t with the launching point at $t+1$.

Then, we can use a set of constraints similar to those used above and the distance from x_R^t to x_L^{t+1} is:

$$z_{RL}^{ee't} = \mu_R^{et}\mu_L^{e't+1}, \tag{50}$$

$$b_{RL}^{it} \leq \sum_{e\in\delta(i)} \mu_R^{et}, \qquad \forall i \in V \tag{51}$$

$$c_{RL}^{it} \leq \sum_{e\in\delta(i)} \mu_L^{et}, \qquad \forall e \in \delta(i), \forall i \in V \tag{52}$$

$$b_{RL}^{it} + b_{RL}^{jt} \geq \mu_R^{et}, \qquad \forall e = (i,j) \in E \tag{53}$$

$$c_{RL}^{it} + c_{RL}^{jt} \geq \mu_L^{et+1}, \qquad \forall e = (i,j) \in E \tag{54}$$

$$b_{RL}^{it} + \sum_{\{j:(i,j)\in E\}} q_{RL}^{jit} = \sum_{\{j:(i,j)\in E\}} q_{RL}^{ijt} + c_{RL}^{it}, \qquad \forall i \in V \tag{55}$$

$$\sum_{e\in E} \mu_L^{et} = 1, \tag{56}$$

$$\sum_{e\in E} \mu_R^{et} = 1, \tag{57}$$

$$d_{RL}^t = \sum_{e,e'\in E} z_{RL}^{ee't} d_{RL}^{ee't}. \tag{58}$$

Constraints (50) model the product of binary variables. Constraints (51) and (52) state that if the rendezvous point (resp. launching point) is not in the edge e, the mothership cannot go (resp. exit) to the end vertices of the edge e. Constraints (53) state that if the mothership leaves the edge e, it must exit via one of the end vertices of e. In the same way, constraints (54) state that if the mothership goes to the edge e, it necessarily must enter to e from one incident vertex of e. Flow conservation constraints (55) ensure that in the route followed by the mothership the number of used incoming edges to each vertex i is equal to the number of used outgoing edges. Constraints (56) and (57) state that, in each stage, the selected points can be only in one edge. Constraints (58) express the total distance between the rendezvous and the launching points at stage t.
Hence, once the distances inside the graph are set with the above two families of constraints, we can state the mathematical programming formulation of the problem as:

$$\min \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\sum_{t\in T}(u^{e_gt}d_L^{e_gt} + v^{e_gt}d_R^{e_gt}) + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\mu^{e_g}d^{e_g} + \sum_{g\in\mathcal{G}}\sum_{e_g,e'_g\in E_g} z^{e_ge'_g}d^{e_ge'_g} + \sum_{t\in T}(d_{RL}^t + d_{LR}^t) \tag{NDRPG-ST}$$

$$\begin{aligned}
\text{s.t.} \quad & (1) - (7), \\
& (39) - (58), \\
& (\text{MTZ}_1) - (\text{MTZ}_2) \text{ or (SEC)}, \\
& (\alpha\text{-E}) \text{ or } (\alpha\text{-G}), \\
& (\text{DCW-t}), \\
& (d_{LR}^{t\mathcal{N}}), (d_{RL}^{t\mathcal{N}}), \\
\|x_L^t - R^{e_g}\| &\leq d_L^{e_gt}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
\|R^{e_g} - L^{e_g}\| &\leq d_g^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
\|R^{e_g} - L^{e'_g}\| &\leq d^{e_ge'_g}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
\|L^{e_g} - x_R^i\| &\leq d_R^{e_gt}, \quad \forall e_g : g \in \mathcal{G}, \forall t \in T, \\
x_L^0 &= orig, \\
x_R^0 &= orig, \\
x_L^{|\mathcal{G}|+1} &= dest, \\
x_R^{|\mathcal{G}|+1} &= dest.
\end{aligned}$$

Again, the objective function has four terms: the first three compute the distances traveled by the drone, while the last one computes the distances traveled by the mothership. Constraints (1)-(7) model the tour made by the drone. Constraints (39)-(58) model the path following by the mothership in the graph. The rest of the constraints are similar to those explained in the model (AMDRPG-ST).

4.3.1. MTZ formulation for the Network Mothership-Drone Routing Problem with Graphs

In order to apply this type of constraints to model connectivity of the routes we have to reformulate the expressions for the distances so that they are not related to stages. In this case, we observe that the distance between launching and rendezvous points in two edges $e = (i, j)$, $e' = (i', j') \in E$, not necessarily distinct, of the graph can be represented as:

$$d_{LR}^{ee'g} = \begin{cases} |\gamma_L^{eg} - \gamma_R^{eg}|\mathcal{L}(e), & \text{if } e = e', \\ \left[b_{LR}^{ig}\gamma_L^{eg} + b_{LR}^{jg}(1-\gamma_L^{eg})\right]\mathcal{L}(e) + \sum_{e''\in\mathcal{N}} q_{LR}^{e''g}\mathcal{L}(e'') + \left[c_{LR}^{i'g}\gamma_R^{e'g} + c_{LR}^{j'g}(1-\gamma_R^{e'g})\right]\mathcal{L}(e'), & \text{if } e \neq e', \end{cases} \tag{$d_{LR}^{g\mathcal{N}}$}$$

where b_{LR}^{ig} (resp. c_{LR}^{ig}) are binary variables that determine from which of the end-points of e (respectively e') one has to account for the distance and q_{LR}^{eg} is a binary variable that is one when the mothership fully traverses the edge e. Furthermore, we need to define binary variables $z_{LR}^{ee'g}$ that model the choice of the model for right definition (first or second formulas in ($d_{LR}^{g\mathcal{N}}$) of the distance in ($d_{LR}^{g\mathcal{N}}$)). All the above arguments give the necessary elements to account for the movement of the mothership on the network $\mathcal{N} = (V, E)$:

$$x_L^g = \sum_{e=(i,j)\in E} \mu_L^{eg}\left[V^i + \gamma_L^{eg}(V^j - V^i)\right] \tag{59}$$

$$x_R^g = \sum_{e=(i,j)\in E} \mu_R^{eg}\left[V^i + \gamma_R^{eg}(V^j - V^i)\right] \tag{60}$$

$$z_{LR}^{ee'g} = \mu_L^{eg}\mu_R^{e'g}, \tag{61}$$

$$b_{LR}^{ig} \leq \sum_{e\in\delta(i)} \mu_L^{eg}, \qquad \forall i \in V \tag{62}$$

$$c_{LR}^{ig} \leq \sum_{e\in\delta(i)} \mu_R^{eg}, \qquad \forall i \in V \tag{63}$$

$$b_{LR}^{ig} + b_{LR}^{jg} \geq \mu_L^{eg}, \qquad \forall e = (i,j) \in E \tag{64}$$

$$c_{LR}^{ig} + c_{LR}^{jg} \geq \mu_R^{eg}, \qquad \forall e = (i,j) \in E \tag{65}$$

$$b_{LR}^{ig} + \sum_{\{j:(i,j)\in E\}} q_{LR}^{jig} = \sum_{\{j:(i,j)\in E\}} q_{LR}^{ijg} + c_{LR}^{ig}, \qquad \forall i \in V \tag{66}$$

$$\sum_{e\in E} \mu_L^{eg} = 1, \tag{67}$$

$$\sum_{e\in E} \mu_R^{eg} = 1, \tag{68}$$

$$d_{LR}^g = \sum_{e,e'\in E} z_{LR}^{ee'g} d_{LR}^{ee'g}, \tag{69}$$

Constraints (59) and (60) parameterize the launching and rendezvous points in the network $\mathcal{N}$ induced by the visit to the graph $g \in \mathcal{G}$. Constraints (61) model the product of binary variables. Constraints (62) and (63) state that if the launching point (resp. rendezvous point) is not in the edge e, the mothership cannot go (resp. exit) to the vertices of the edge e. Constraints (64) state that if the mothership leaves the edge e, it must exit from one of the incident vertices to e. In the same way, constraints (65) state that if the mothership leaves the edge e', it necessarily must enter to e from one incident vertex of e. Flow conservation constraints (66) ensure that, in the route to be defined, the number of incoming edges to each vertex i is equal to the number of outgoing edges. Constraints (67) and (68) state that, for the visit to the graph $g \in \mathcal{G}$, launching and rendezvous points can be only in one edge. Constraints (69) returns the total distance traveled by the drone on the graph $g \in \mathcal{G}$.

Similarly, the distance covered by the mothership along the path on the network $\mathcal{N}$, from the rendezvous point $x_R^g \in e$, after the visit to $g \in \mathcal{G}$ to the next launching point $x_L^{g'} \in e'$ (to go to the graph $g' \in \mathcal{G}$), can be modeled using the following definition of distance:

$$d_{RL}^{ee'gg'} = \begin{cases} |\gamma_R^{eg} - \gamma_L^{eg'}|\mathcal{L}(e), & \text{if } e = e', \\ \left[b_{RL}^{ig}\gamma_R^{eg} + b_{RL}^{jg}(1-\gamma_R^{eg})\right]\mathcal{L}(e) + \sum_{e''\in\mathcal{N}} q_{RL}^{e''gg'}\mathcal{L}(e'') + \left[c_{RL}^{i'g'}\gamma_L^{e'g'} + c_{RL}^{j'g'}(1-\gamma_L^{e'g'})\right]\mathcal{L}(e'), & \text{if } e \neq e'. \end{cases} \tag{$d_{RL}^{g\mathcal{N}}$}$$

In this case, the binary variable $z_{RL}^{ee'gg'}$ links the rendezvous point at g with the launching point at g'. Then, we can use a set of constraints similar to those used above and the distance from x_R^g to $x_L^{g'}$ is:

$$z_{RL}^{ee'gg'} = \mu_R^{eg}\mu_L^{e'g'}, \tag{70}$$

$$b_{RL}^{ig} \leq \sum_{e\in\delta(i)} \mu_R^{eg}, \qquad \forall i \in V \tag{71}$$

$$c_{RL}^{ig} \leq \sum_{e\in\delta(i)} \mu_L^{eg}, \qquad \forall i \in V \tag{72}$$

$$b_{RL}^{ig} + b_{RL}^{jg} \geq \mu_R^{eg}, \qquad \forall e = (i,j) \in E \tag{73}$$

$$c_{RL}^{ig} + c_{RL}^{jg} \geq \mu_L^{eg}, \qquad \forall e = (i,j) \in E \tag{74}$$

$$b_{RL}^{ig} + \sum_{\{j:(i,j)\in E} q_{RL}^{jigg'} = \sum_{\{j:(i,j)\in E\}} q_{RL}^{ijgg'} + c_{RL}^{ig'}, \qquad \forall i \in V \tag{75}$$

$$d_{RL}^{gg'} = \sum_{e,e'} z_{RL}^{ee'gg'} d_{RL}^{ee'gg'}. \tag{76}$$

Hence, once the distances inside the graph are set with the above two families of constraints, we can state the mathematical programming formulation of the problem as:

$$\min \quad \sum_{g\in E_g} (u^{e_g} d_L^{e_g} + v^{e_g} d_R^{e_g}) + \sum_{e_g\in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g, e'_g \in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{g\in E_g} d_{LR}^g + \sum_{g,g'\in E_g} d_{RL}^{gg'} w^{gg'} \tag{NDRPG-MTZ}$$

$$\begin{aligned} \text{s.t.} \quad & (1) - (7), \\ & (59) - (76), \\ & (\text{MTZ}_1) - (\text{MTZ}_2) \text{ or (SEC)}, \\ & (\alpha\text{-E}) \text{ or } (\alpha\text{-G}), \\ & (\text{DCW-g}), \\ & (d_{LR}^{g\mathcal{N}}), (d_{RL}^{g\mathcal{N}}), \\ & \|x_L^g - R^{e_g}\| \leq d_L^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \\ & \|R^{e_g} - L^{e_g}\| \leq d_g^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \\ & \|R^{e_g} - L^{e'_g}\| \leq d^{e_g e'_g}, \quad \forall e_g : g \in \mathcal{G}, e'_g, \\ & \|L^{e_g} - x_R^g\| \leq d_R^{e_g}, \quad \forall e_g : g \in \mathcal{G}, \\ & x_L^0 = orig, \\ & x_R^0 = orig, \\ & x_L^{|\mathcal{G}|+1} = dest, \\ & x_R^{|\mathcal{G}|+1} = dest. \end{aligned}$$

5. Strengthening the formulations of MDRPG

The different models that we have proposed include in one way or another big-M constants. We have defined different big-M constants along this work. In order to strengthen the formulations we provide tight upper and lower bounds for those constants. In this section we present some results that adjust them for each one of the the models.

Big M constants bounding the distance from the launching / rendezvous point on the path followed by the mothership to the rendezvous / launching point on the target graph $g \in \mathcal{G}$

- AMDRPG. To linearize the first term of the objective function in AMDRPG, we define the auxiliar non-negative continuous variables $p_L^{e_g t}$ and (resp. $p_R^{e_g t}$) to model the product by including the following constraints:

$$\begin{aligned} p_L^{e_g t} &\geq m_L^{e_g} u^{e_g t}, \\ p_L^{e_g t} &\leq d_L^{e_g} - M_L^{e_g t}(1 - u^{e_g t}). \end{aligned}$$

  The best upper bound $M_R^{e_g t}$ or $M_L^{e_g t}$ that we can consider is the full diameter of the data, that is the maximum distance between every pair of vertices of the graphs $g \in \mathcal{G}$, in input the data, i.e., every launching or rendezvous point is inside the circle whose diametrically opposite points are the explained below.

$$M_R^{e_g t} = \max_{\{v \in V_g, v' \in V_{g'} : g, g' \in \mathcal{G}\}} \|v - v'\| = M_L^{e_g t}.$$

  On the other hand, the minimum distance in this case can be zero. This bound is attainable whenever the launching or the rendezvous points of the mothership is the same that the rendezvous or launching point on the target graph $g \in \mathcal{G}$.

- NMDRPG. In this case, the best upper bounds for $M_R^{e_g t}$ or $M_L^{e_g t}$ is the maximum distance between the polygonal chain $\mathcal{P}$ or the graph $\mathcal{N}$ and any of the target graphs $g \in \mathcal{G}$:

$$M_R^{e_g t} = \max_{\{v \in V_g, w \in \mathcal{N}\}} \|v - w\| = M_L^{i_g t}.$$

  On the other hand, the minimum distance can be computed by taking the closest points between the graph g and the network $\mathcal{N}$:

$$m_R^{e_g t} = \min_{\{v \in V_g, w \in \mathcal{N}\}} \|v - w\| = m_L^{e_g t}.$$

Bounds on the big M constants for the distance from the launching point to the rendezvous points for the MTZ/SEC formulations in AMDRPG

We can compute a tighter upper bound for the distance $d_{RL}^{gg'}$ between each pair of graphs g, g' for the constraints obtained by the linearization of its product:

$$\begin{aligned} p^{gg'} &\geq m_{RL}^{gg'} d_{RL}^{gg'}, \\ p^{gg'} &\leq d_{RL}^{gg'} - M_{RL}^{gg'}(1 - w^{gg'}). \end{aligned}$$

This upper bound $M_{RL}^{gg'}$ is given by the diameter of $g \cup g'$:

$$M_{RL}^{gg'} = \max_{\{v \in V_g, v' \in V_{g'}\}} \|v - v'\|.$$

Bounds on the big M constants for the distance from the launching to the rendezvous points on the target graph $e \in \mathcal{G}$.

When the drone visits a graph g, it has to go from one edge e_g to another edge e'_g depending on the order given by $z^{e_g e'_g}$. This fact produces another product of variables linearized by the following constraints:

$$\begin{aligned} p^{e_g e'_g} &\geq m^{e_g e'_g} d_{RL}^{gg'}, \\ p^{e_g e'_g} &\leq d^{e_g e'_g} - M^{e_g e'_g}(1 - z^{e_g e'_g}). \end{aligned}$$

Since we are taking into account the distance between two edges $e = (B^{e_g}, C^{e_g}), e' = (B^{e'_g}, C^{e'_g}) \in E_g$, the maximum and minimum distances between their vertices give us the upper and lower bounds:

$$\begin{aligned} M^{e_g e'_g} &= \max\{\|B^{e_g} - C^{e'_g}\|, \|B^{e_g} - B^{e'_g}\|, \|C^{e_g} - B^{e'_g}\|, \|C^{e_g} - C^{j_g}\|\}, \\ m^{e_g e'_g} &= \min\{\|B^{e_g} - C^{e'_g}\|, \|B^{e_g} - B^{e'_g}\|, \|C^{e_g} - B^{e'_g}\|, \|C^{e_g} - C^{e'_g}\|\}. \end{aligned}$$

Bounds on the big M constants for the distance covered by the mothership on the polygonal for the PMDRPG *model during one drone operation.*

In the case of PMDRPG, we can also set tighter upper bounds for the distance covered by the drone inside the polygonal during an operation that starts in e and finishes at e' (or vice versa) (see (23) and (24)). This is clearly bounded from above by the total length of the line segments where the mothership is located.

$$M_{LR}^{ee't} = M_{RL}^{ee't} = \begin{cases} \mathcal{L}(e), & \text{if } e = e', \\ \sum_{e<e''<e'} \mathcal{L}(e'') & \text{if } e < e', \\ \sum_{e'<e''<e} \mathcal{L}(e'') & \text{if } e > e'. \end{cases}$$

Bounds on the big M constants for the distance covered by the drone during an operation for all the models by stages

To link the drone operation with the mothership travel in the models by stages, we have defined the constraint (DCW-t) that includes the M:

$$\left(\sum_{e_g\in E_g} u^{e_g t} d_L^{e_g t} + \sum_{e_g, e'_g\in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{e_g\in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g\in E_g} v^{e_g t} d_R^{e_g t}\right) / v_D \leq d_{RL}^t / v_M + M(1 - \sum_{e_g\in E_g} u^{e_g t})$$

To obtain this upper bound M we add to the length of the graph $\mathcal{L}(g)$ the big-Ms computed for $u^{e_g t}$ and $v^{e_g t}$, that is, $M_L^{e_g t}$ and $M_R^{e_g t}$, and the maximum distances that can be employed by the drone to move from one edge to another one:

$$M = \mathcal{L}(g) + M_L^{e_g t} + M_R^{e_g t} + \sum_{e_g, e'_g\in E_g} M^{e_g e'_g}.$$

6. A Matheuristic for the Mothership-Drone Routing Problem with Graphs

This section is devoted to present our matheuristic approach to address the solution of the MDRPG. Our motivation comes from the fact that the exact methods presented in the previous section are highly time demanding. Alternatively, the matheuristic provides good quality solution in limited computing times. First, we focus on the case in which the mothership can move freely on the continuous space, namely AMDRPG. The basic idea of the algorithm is to determine first the sequence of visits to the target graphs in $\mathcal{G}$. This route is obtained replacing each graph in $\mathcal{G}$ by its centroid and then solving a TSP over the neighborhoods of those centroids, namely a XPPN (see [23]). Then, given such an order of visits, the launching/rendezvous points where the mothership must stop, are determined by solving a AMDRPG problem, but limited to one single graph per time, following the sequence previously computed. Successively, the solution so obtained is improved by providing it to the solver as initial partial solution and by solving, once more, another AMDRPG problem, where now we fix the launching/rendezvous points but leave free the variables w that give the sequence of visits to the targets graphs, i.e., without fixing the order of visits. If the solution so obtained consists in a different order of visit of the graphs with respect to the previous one, the procedure is repeated to generate a new feasible solution. The stopping criterion is to find a solution that does not change with respect to the one obtained in the previous cycle of the algorithm. In the following, we present the pseudo-code of this algorithm:

STEP 1 (Centroids of the graphs)
- Let *orig* be the origin/destination of the mothership tour (orange point with label 0 in Figure 4 [a]).
For each graph $g \in \mathcal{G}$:
- identify its centroid c_g and consider its neighborhood defined as the circle $\rho(c_g, 2)$ (red points in Figure 4 [a]).

STEP 2 (Order of visit of the graphs) Determine an order of visit for the graphs in $\mathcal{G}$ by solving the XPPN of the mothership over the set of the neighborhoods associated with the centroids of those graphs (blue tour 0, 1, 2, 3, 4, 0 in Figure 4 [a]).

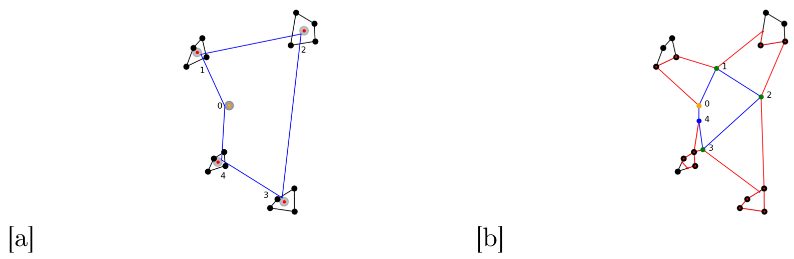

Figure 4: Illustrative example

STEP 3 (launching/rendezvous points location) Let $\bar{w}_{gg'}$, $\forall g, g' \in \mathcal{G}$ be the optimal values of the variables $w_{gg'}$ generated by STEP 2.
Following this order of visit, set the launching point for the first graph as the depot, then solve the resulting AMDRPG limited to the first graph.
Repeat the same procedure for the remaining graphs to be visited, by solving AMDRPG on one single graph per time, by fixing as launching point of the current graph the rendezvous point of the previous graph.

STEP 4 (Solution update) Let $\bar{z}$ be the solution obtained by STEP 3, consisting of the tour of the drone on each target (Figure 4 [b]), and $\bar{x}_L^g$, $\bar{x}_R^g$ $\forall g \in \mathcal{G}$ the associated launching/rendezvous points (green and blue points in Figure 4 [b]).
Solve the model AMDRPG with these launching/rendezvous points but leaving free the $w_{gg'}$ $\forall g, g' \in \mathcal{G}$ variables and providing to the solver $\bar{z}$ as initial partial solution.

STEP 5 Let $\hat{z}$ be the updated solution obtained by STEP 4 (again Figure 4 [b] for this illustrative example) and $\hat{w}_{gg'}$ the associated order of visit of the graphs.
If the $\hat{w}_{gg'} \neq \bar{w}_{gg'}$ repeat from STEP 3, otherwise stop.

Note that in the example shown in Figure 4 we solved the AMDRPG model by imposing that at least a given percentage of each target must be visited. For this reason the solution consists in drone's missions which visit only part of each graph, as represented by the red segments in Figure 4 [b]. Moreover, for this example, most of the launching/rendezvous points coincide. More precisely, considering Figure 4 [b], the drone starts from the origin and visits part of the first target. The mothership retrieves it at the point labelled 1. From the same point the drone starts the second mission to visit part of the second target and then reaches the mothership at the point labelled 2. From that point the drone starts for visiting the third target meeting the mothership at the point labelled 3. From this latter point the last mission of the drone starts and ends on the mothership at the point labelled 4. Then the mothership with the drone go back to the origin. In this example the solution obtained at STEP 4, by fixing the launching/rendezvous points but not the order of visit, does not change with respect to the one obtained at STEP 3. Thus the procedure stops after the first iteration and provides the solution shown in Figure 4 [b]. The exact resolution of AMDRPG can be hard expecially on large instances. For this reason for each call of the solver within the heuristic procedure, we set a maximum number of solutions that the solver can generate before stopping.
The procedure adopted for the NMDRPG problem consists in the same steps of the matheuristic presented above for the AMDRPG problem but, at STEP 4, the partial solution, given in input to the solver, includes also the values of the variables modelling the routing of the truck on the graph.

7. Experimental results

In this section we discuss the experimental results obtained testing the formulations presented in Section 4 and the matheuristic procedure proposed in Section 6 on different sets of instances. In particular, we generated two sets of instances of the AMDRPG problem. The first one consists of targets, to be visited by the drone, that are represented by grid graphs, while the second one involves a different typology of targets, that is, Delaunay graphs. For both typologies we generated 5 instances of 10 graphs each, with different cardinality of the set of nodes. More precisely, each instance is composed of 3 graphs with 4 nodes, 3 graphs of 6 nodes, 3 graphs of 8 nodes and 1 graph of 10 nodes. Moreover, we assumed that the drone's speed is twice that of the mothership and that a percentage equal to 80% of each target must be

visited by the drone.
In order to locate in the space the 10 graphs of a single instance, we considered a square of side 100 units. Then we divided the original square in subsquares of side 5 and we randomly selected among them the locations for the ten target graphs of the instance. The generation procedure of the single graph in a selected square, depends on the graph typology. As regards grid graphs, the single subsquare, like the one represented in Figure 5, is further partitioned in subsquares of side $\frac{5}{n}$ where n is the cardinality of the set of nodes of the graph to build. Two opposite corner subsquares have been considered (like V_1 and V_2 in Figure 5) and one point inside each of them was randomly selected (black points in the subsquares V_1 and V_2 in Figure 5). Then, a rectangle whose diagonal joins these two points has been built (the black dotted one in Figure 5). A grid of n points has been identified by locating $\frac{n}{m}$ equally spaced points on the two sides square (the red ones and the two original points in black in Figure 5), where m is randomly selected in the set of divisors of the number of points of the graph. The links of the graphs connect each point to its adjacent ones lying on the same side and with the one located on the opposite side of the square. Let $width_x$ and $width_y$ be the lengths of these edges as show in Figure 5. In order to perturb the coordinates of these points, we randomly added a value, ranging between $-\frac{width_x}{3}$ and $\frac{width_x}{3}$ to the x coordinate and between $-\frac{width_y}{3}$ and $\frac{width_y}{3}$ to the y coordinate, always imposing that the perturbed point still belongs to the square. The resulting grid graph is obtained connecting the same pairs of points but with perturbed coordinates (blue graph in Figure 5).
As regards the instances related to Delaunay graphs, we adopted the same procedure as for the grid ones, for selecting their locations in the space. Then, for each randomly selected subsquare, given the set of n nodes of the graph to be generated, a Delaunay triangulation of them is computed by using the Python class scipy.spatial.Delaunay (see [26] for further details).

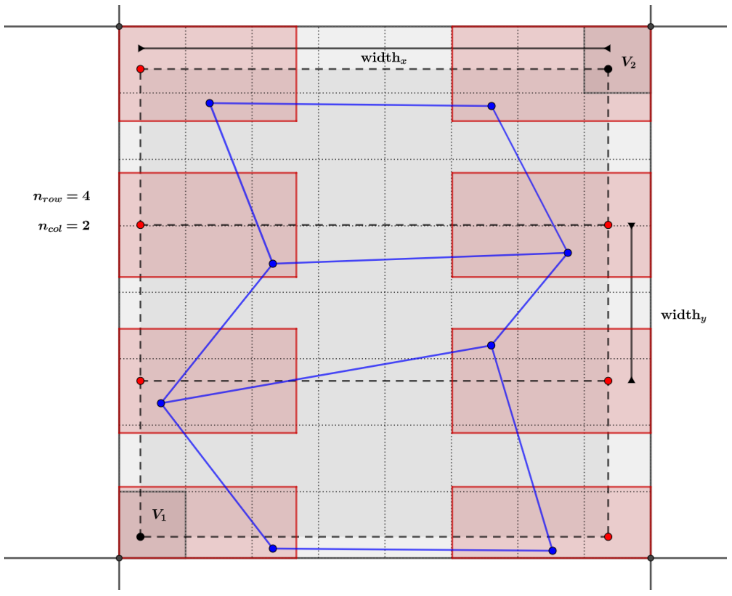

Figure 5: Example of generation of a grid graph

We run the three formulations proposed in Section 4 for the AMDRPG problem (Stages, MTZ and SEC) with two different commercial solvers, Cplex 12.8 and Gurobi 9.03, called by means of Python, by setting a time limit for each run equal to 3600 sec. In Table 1 we reported the results obtained, in terms of average, minimum and maximum percentage with both solvers, on the instances consisting of grid graphs. First, we can observe that Gurobi has better performances with respect to Cplex for all the instances. Moreover, for this set of instances, the SEC formulation is the best one among the three proposed, as the associated average gap is equal to 0.61% and also the minimum and the maximum percentage gap are smaller than the ones associated with the Stages and the MTZ formulations.

Similarly, Table 2 summarizes the results obtained on the instances with Delaunay graphs. Also in this case the Gurobi performances are better than Cplex. However, among the three formulations, the MTZ one provides the best results in terms of average and maximum percentage gap.
In order to test the performances of the matheuristic proposed in Section 6, we coded it in Python

Table 1: Comparison between formulations for grid instances

Gap %	Average		Min		Max	
Solver	Cplex	Gurobi	Cplex	Gurobi	Cplex	Gurobi
Formulation						
Stages	0,87	0,87	0,85	0,84	0,88	0,88
MTZ	0,66	0,62	0,59	0,58	0,72	0,65
SEC	0,65	0,61	0,59	0,57	0,70	0,64

Table 2: Comparison between formulations for Delauney instances

Gap %	Average		Min		Max	
Solver	Cplex	Gurobi	Cplex	Gurobi	Cplex	Gurobi
Formulation						
Stages	0,91	0,91	0,90	0,89	0,93	0,93
MTZ	0,78	0,74	0,74	0,70	0,82	0,79
SEC	0,77	0,75	0,73	0,69	0,82	0,81

and we run it on the same sets of instances (Grid and Delaunay) on which the three formulations have been solved. Table 3 reports for each instance, numbered from 0 to 4, distinguishing between Grid and Delauney, respectively, the best objective function provided by the best formulation, the objective function provided by the matheuristic and the associated CPU time. As already noticed from Table 2, for grid graph instances, SEC formulation has the best behaviour, with the exception of the instance number 3 for which the MTZ provides a smaller value of the objective function. As for the Delaunay graph instances, MTZ is the best formulation, but also in this case there is an exception on the instance number 2 for which SEC formulation returns a smaller value of the objective function.
The results show that the matheuristic returns a solution with value of the objective function that is higher than the one provided by the SEC formulation on grid instances. However, these values are smaller than the ones provided by the Stages and the MTZ formulations. Moreover, the saving in terms of resolution time is very significant as the maximum CPU time is less than 1 minute. As regards the Delaunay instances, the matheuristic performances are even better, as it finds a solution that is better than the best one provided by the MTZ formulation and in a resolution time that is at most 28 minutes.

Table 3: Heuristic performances

#	**Grid**			**Delauney**		
	Best Obj	Obj Heuristic	CPU Time	Best Obj	Obj Heuristic	CPU Time
0	1087,87	1117,83	50,99	947,01	934,46	52,49
1	1100,38	1319,64	24,64	986,22	938,68	72,73
2	1350,67	1126,35	46,06	888,48	865,66	1073,80
3	1218,66	1476,36	27,18	1249,69	1154,62	1703,33
4	1297,77	1424,37	40,91	1239,93	1184,67	81,15

We performed a second set of experiments by observing that, even if there are small differences between the SEC and the MTZ formulations depending on the type of instances, their performances are comparable. Thus, in the rest of the tests we focused on the MTZ formulation. We compared its performances, with or without providing the initial solution found by the matheuristic, on a set of larger instances. More precisely, we generated 20 instances with targets represented by grid graphs and 20 instances with targets represented by Delauney graphs. The instances of each typology are split in 4 groups of 5 instances each, consisting respectively of 5, 10, 15 and 20 targets to be visited. In each instance the same percentage of graphs (20%) has respectively 4, 6, 8, 10 and 12 nodes. Moreover, we assumed that the origin coincides with the destination in all instances and we randomly generated with uniform distribution between 0 and 1, two values representing the percentage of each edge and of each graph to be visited. As regards the speeds, we set the speed of the drone three times the one of the mothership. We run the MTZ formulation by adopting Gurobi, setting a time limit of 7200 sec. for each instance. On the same instances also the matheuristic has been applied. Note that, in order to define a stopping rule for the exact resolution of the AMDRPG model within the matheuristic procedure (STEP 3 and STEP 4), we set the maximum number of solutions generated by the solver equal to five. For each instance, the solution provided by the matheuristic has been then used to initialize the exact resolution of the MTZ formulation in order to try to speed up the resolution process. Table 4 shows the results of the comparison between the exact resolution of the formulation with and without initialization. In the first column, named List, we report the size of the instances in terms of number of targets to be visited (0, 1, 2 and 3 identifies instances respectively with 5, 10, 15 and 20 graphs).The second column refers to the two variants of the model, that is, a given percentage of each edge of the targets (e) or a given percentage of each target (g) must be visited by the drone. The other columns report respectively the average percentage gap of the solutions found within the time limit starting from the initial solution provided by the matheuristic, the average running time of the matheuristic and the average percentage gap of the solutions found within the time limit without initialization. These information are reported for both Grid and Delauney instances.

Table 4: Comparison between exact resolution with and without initialization

		Grid			Delauney		
List	%	% Gap (i)	Time_h	% Gap (ni)	% Gap (i)	Time_h	% Gap (ni)
	e	0.72	105.12	0.73	0.78	154.92	0.74
	g	0.55	58.92	0.54	0.62	92.64	0.67
	e	0.76	241.99	0.76	0.80	314.69	0.79
	g	0.71	182.61	0.70	0.74	353.04	0.75
	e	0.76	367.69	0.76	0.80	447.61	0.80
	g	0.71	326.49	0.72	0.76	429.16	0.76
	e	0.75	481.68	0.74	0.80	514.98	0.76*
	g	0.71	492.27	0.70	0.77	582.90	0.77

From Table 4 we can notice that in most of the cases the average gaps associated with the solution found within the time limit, with and without initialization by the solution found by the matheuristic, are the same or very close (note that in the last column the $*$ indicates that only one instance has been solved within the time limit). As regards the running time of the matheuristic, we can see also from the boxplots in Figure 6, that it increases with the number of targets to be visited both for Grid and Delaunay instances. Considering the model variants based on the minimum percentage of each edge or each graph to visit, we can observe that for Grid instances the average running time of the model imposing a minimum percentage of each edge to be visited, is greater than the one associated with the other variant, with the exception of the instances of biggest size (List=3).
The boxplots in Figure 7 represent the percentage gap of the solution provided by the matheuristic with respect to the one provided by the exact resolution of the MTZ model within the time limit, with initialization by the solution found by the matheuristic. From them we can notice that the gap increases with the size both for Grid and Delaunay instances but it is always less than 0.5%. Figure 8 shows the percentage gap of the solution provided by the exact resolution of the MTZ formulation within the time limit without the initialization, with respect to the one found with the initialization. This gap is very close to 0 both for Grid and Delaunay instances. Only for the biggest size we observe values ranging between 0.1% and 0.6%. These observations suggest that, even if the initialization of the model by the solution provided by the matheuristic does not speed up the convergence to the optimal solution, the matheuristic provides solutions of very good quality. Indeed, it generates in less than 10 minutes solutions that are very close to the ones provided by the model within 2 hours.

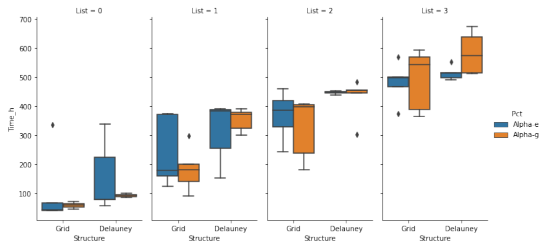

Figure 6: Matheuristic running time

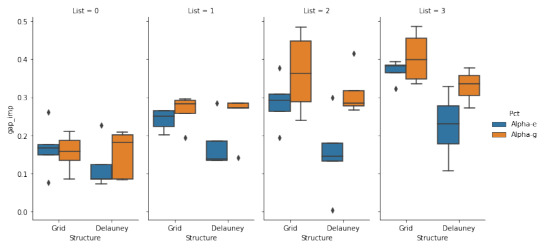

Figure 7: Matheuristic improved gap

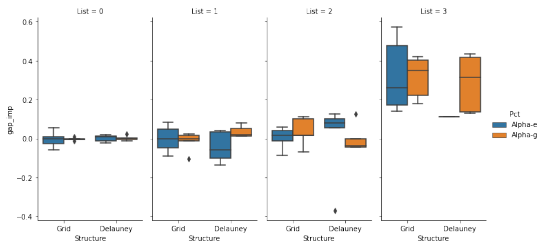

Figure 8: Improved gap of MTZ formulation with and without initialization

As regards the NMDRPG problem, we generated three sets of instances with targets represented by grid graphs considering different structures of the polygonal network where the mothership can move. In particular, we defined a first set of instances where the mothership network is represented by a graph of

6 nodes with a tree structure with origin of the path of the base vehicle different from the destination. A second set of instances involving a mothership network consisting in a complete graph of 4 vertices with origin of the path of the base vehicle different from the destination. A third set of instances characterized by star graphs of 7 nodes representing the mothership network, where the origin coincides with the destination and it is located at the centre of the star. We generated 10 instances for each of these three classes, 5 of them with 5 targets and 5 with 10 targets to be visited. Moreover, as for the AMDRPG, for each of these 10 instances we randomly generated two values representing the percentage of each edge and of each graph that must be visited by the drone. We run on these sets of instances both Stages and MTZ formulations. Table 5 summarizes the results obtained comparing them. The first column identifies the size of the instances, similarly to Table 4, (0 for instances with 5 targets and 1 for instances with 10 targets). The second column distinguishes between minimum percentage of each edge (e) or of each graph (g) to be visited by the drone. The remaining columns refer to the three different class of instances described above (1 for the networks with a tree structure, 2 for complete networks and 3 for start networks). For each of these sets of instances the average percentage gap of the solutions found within the time limit of 7200 sec. by the two formulations (Stages and MTZ) is reported.

Table 5: Comparison between formulations of NMDRPG

	Net Struct	1		2		3	
List	%	Stages	MTZ	Stages	MTZ	Stages	MTZ
	e	0.89	0.33	0.88	0.24	0.87	0.39
	g	0.86	0.29	0.89	0.18	0.90	0.42
	e	0.92	0.43	0.92	0.33	0.92	0.46
	g	0.91	0.36	0.92	0.23	0.92	0.39

We can observe that for each class of instances and model variants, based on the percentage of each edge or each graph to be visited, the MTZ formulation performs better than the Stages one. In all the cases the percentage average gap associated with the MTZ formulation is one third or half of that associated with the Stages formulation. For this reason, in the following tests, related to the comparison between the exact resolution of the NMDRPG model with and without the initialization by the solution found by the matheuristic, we focused only on the MTZ formulation.
Table 6 summarizes the results of this comparison distinguishing again between the different network structures (columns labelled 1, 2 and 3), the different size (rows labelled 0 and 1) characterizing the instances and model variants (minimum percentage of each edge (e) or each graph (g) to be visited). For each combination of network structure, size and model variant we reported the average percentage gap with initialization (% Gap (i)), the solution time of the matheuristic (T_h) and the average percentage gap without initialization by the solution found by the matheuristic (% Gap (ni))

Table 6: Comparison between exact resolution with and without initialization of NMDRPG

	Net Struct	1			2			3		
List	%	% Gap (i)	T_h	% Gap (ni)	% Gap (i)	T_h	% Gap (ni)	% Gap (i)	T_h	% Gap (ni)
	e	0.32	109.96	0.33	0.24	207	0.24	0.39	177.57	0.39
	g	0.30	110.92	0.29	0.18	163.36	0.18	0.45	149.68	0.42
	e	0.48	1030.64	0.43	0.39	802.3	0.33	0.53	770.05	0.46
	g	0.33	479.36	0.36	0.35	639.09	0.23	0.42	689.51	0.39

We can observe that, similarly to the AMDRPG problem, the average gaps associated with the solution found within the time limit, with and without initialization by the solution found by the matheuristic, are very close. Considering the average running time we can notice that the NMDRPG problem is more challanging to be solved with respect to the AMDRPG. It increases very fast with the size of the instances especially for the case in which the network where the mothership moves has a tree structure. Moreover, as for the Grid instances in the continuous case, the model variant imposing a minimum percentage of each edge to be visited takes more time to be solved.
The boxplots showed in Figure 9 report the percentage gap of the solution provided by the matheuristic with respect to the one provided by the exact resolution of the MTZ model within the time limit, with initialization by the solution found by the matheuristic. We can notice that, excluding the outliers, this gap ranges between 0% and 0.7% and its lowest values are observed for the instances in which the network where the mothership moves has a star structure (green boxplots). From the previous observations, similarly to the AMDRPG, we can conclude that the behaviour of the matheuristic is very good in terms of quality of the solutions provided, even if the initialization of the MTZ model does not help in speeding up the convergence to the optimal solution.

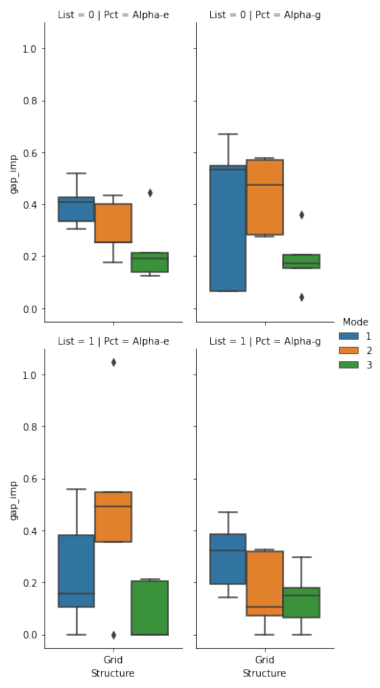

Figure 9: Matheuristic improved gap for NMDRPG

8. Concluding remarks

This papers has analyzed the coordination problem that arises between a mothership vehicle and a drone that must adjust their routes to minimize travel distances while visiting a set of targets modeled by graphs. We have presented exact formulations for different versions of the problem depending on the constraints imposed to the mothership movement (free on a continuous space or on a given network). Our computational results show that the considered problem is rather hard and only small to medium size problems can be solved to optimality. Additionally, we have proposed a matheuristic algorithm, applicable to all the versions of the problem with minimum changes, that provides acceptable feasible solutions in short computing time; so that it is a good alternative to the exact methods.
Further research in this topic includes the coordination of the operations of several drones with a mothership, the possibility of visiting more than one target per operation and combinations of both cases. These problems being very interesting are beyond the scope of this paper and will be the focus of a follow up paper.

Acknowledgements

This research has been partially supported by Spanish Ministry of Education and Science/FEDER grant number MTM2016-74983-C02-(01-02), projects FEDER-US-1256951, CEI-3-FQM331 and *NetmeetData*: Ayudas Fundación BBVA a equipos de investigación científica 2019, and by University of Rome, Sapienza grant number: RM11916B7F962975.

References

[1] Amorosi, L., Caprari, R., Crainic, T., Dell'Olmo, P., and Ricciardi, N. (2020). CIRRELT-2020-17 An Integrated Routing-Scheduling Model for a Hybrid UAV-Based Delivery System. CIRRELT.

[2] Amorosi, L., Chiaraviglio, L., and Galan-Jimenez, J. (2019). Optimal energy management of uav-based cellular networks powered by solar panels and batteries: Formulation and solutions. IEEE Access, Vol. 7:53698–53717.

[3] Campbell, J. F., Corberán, Á., Plana, I., and Sanchis, J. M. (2018). Drone arc routing problems. Networks, 72(4):543–559.

[4] Campbell, J. F., Sweeney, D., and Zhang, J. (2017). Strategic design for delivery with trucks and drones. Computer Science.

[5] Carlsson, J. G. and Song, S. (2018). Coordinated logistics with a truck and a drone. Management Science, 64(9):4052–4069.

[6] Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., Dell'olmo, P., Lo Mastro, A., Natalino, C., and Monti, P. (2019a). Minimum Cost Design of Cellular Networks in Rural Areas with UAVs, Optical Rings, Solar Panels, and Batteries. IEEE Transactions on Green Communications and Networking, 3(4):901–918.

[7] Chiaraviglio, L., Amorosi, L., Malandrino, F., Chiasserini, C. F., Dell'Olmo, P., and Casetti, C. (2019b). Optimal Throughput Management in UAV-based Networks during Disasters. INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019, pages 307–312.

[8] Chow, J. Y. (2016). Dynamic UAV-based traffic monitoring under uncertainty as a stochastic arc-inventory routing policy. International Journal of Transportation Science and Technology, 5(3):167–185.

[9] Chung, S. H., Sah, B., and Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. Computers & Operations Research, 123:105004.

[10] Di Puglia Pugliese, L. and Guerriero, F. (2017). Last-Mile Deliveries by Using Drones and Classical Vehicles BT - Optimization and Decision Science: Methodologies and Applications. pages 557–565, Cham. Springer International Publishing.

[11] Dille, M. and Singh, S. (2013). Efficient aerial coverage search in road networks. AIAA Guidance, Navigation, and Control (GNC) Conference.

[12] Garone, E., Naldi, R., Casavola, A., and Frazzoli, E. (2010). Cooperative mission planning for a class of carrier-vehicle systems. Proceedings of the IEEE Conference on Decision and Control, pages 1354–1359.

[13] Jimenez, J. G., Chiaraviglio, L., Amorosi, L., and Blefari-Melazzi, N. (2018). Multi-Period Mission Planning of UAVs for 5G Coverage in Rural Areas: A Heuristic Approach. Proceedings of the 2018 9th International Conference on the Network of the Future, NOF 2018, pages 52–59.

[14] Li, M., Zhen, L., Wang, S., Lv, W., and Qu, X. (2018). Unmanned aerial vehicle scheduling problem for traffic monitoring. Computers and Industrial Engineering, 122:15–23.

[15] Mathew, N., Smith, S. L., and Waslander, S. L. (2015). Planning Paths for Package Delivery in Heterogeneous Multirobot Teams. IEEE Transactions on Automation Science and Engineering, 12(4):1298–1308.

[16] Moshref-Javadi, M. and Lee, S. (2017). Using drones to minimize latency in distribution systems. In IE Annual Conference, Proceedings, pages 235–240.

[17] Mourelo Ferrandez, S., Harbison, T., Weber, T., Sturges, R., and Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. Journal of Industrial Engineering and Management; Vol 9, No 2 (2016)DO - 10.3926/jiem.1929.

[18] Oh, H., Kim, S., Tsourdos, A., and White, B. A. (2014). Coordinated road-network search route planning by a team of UAVs. International Journal of Systems Science, 45(5):825–840.

[19] Oh, H., Shin, H. S., Tsourdos, A., White, B. A., and Silson, P. (2011). Coordinated Road Network Search for Multiple UAVs Using Dubins Path BT - Advances in Aerospace Guidance, Navigation and Control. pages 55–65, Berlin, Heidelberg. Springer Berlin Heidelberg.

[20] Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. Networks, Vol. 72:1–48.

[21] Poikonen, S. and Campbell, J. (2020). Future directions in drone routing research. Networks, pages 1–11.

[22] Poikonen, S. and Golden, B. (2020). Multi-visit drone routing problem. Computers and Operations Research, 113:104802.

[23] Puerto, J. and Valverde, C. (2020). Routing for unmanned aerial vehicles: Touring dimensional sets. arXiv.

[24] Tokekar, P., Hook, J. V., Mulla, D., and Isler, V. (2016). Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. IEEE Transactions on Robotics, 32(6):1498–1511.

[25] Trotta, A., Andreagiovanni, F. D., Di Felice, M., Natalizio, E., and Chowdhury, K. R. (2018). When UAVs Ride A Bus: Towards Energy-efficient City-scale Video Surveillance. Proceedings - IEEE INFOCOM, 2018-April:1043–1051.

[26] Virtanen, P., Gommers, R., Oliphant, T. E., and et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 17(3):261–272.

[27] Wen, T., Zhang, Z., and Wong, K. K. L. (2016). Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation. PLoS ONE, 11(5).