Thank you for your presentation, Mr Campbell.

Good afternoon, my name is Carlos Valverde Martín and now I am going to present my joint work with Justo Puerto and Lavinia Amorosi, whose name is The Mothership and Drone Routing Problem with Graphs.

Here I show the contents of my talk:

* Firstly, we talk about previous works in literature related to the use of drones.
* Secondly, we describe the problem, what is the starting point and how we have extended this point.
* Thirdly, we give some Mixed-Integer-NonLinear-Programming formulations for the All-Terrain Mothership and Drone Routing Problem with Graphs (AMDRPG)
* Then, we expose an matheuristic that allows to obtain feasible solutions to the problem.
* Finally, we discuss some computational experiments and their results, and we give some final conclusions that we obtain when we try to solve this problem.

Regarding Arc Routing Problem with drones, the literature is limited.

* We can highlight the work made by Campbell, Corberan, Plana and Sanchis, namely, drone arc routing problems that shows the relation with the Intermittent Cutting Problem.
* We also have to mention the work of Tokekar, Hook, Mulla and Isler, where they study the path planning problem of a system composed by a ground robot and one drone in precision agriculture.
* We can see the future directions in drone routing research in the work of Campbell and Poikonen.

But our starting point was The Mothership and Drone Routing Problem (MDRP), defined by Golden and Poikonen in 2018.

In this problem, we have an origin that coincides with the destination and it is represented by black square and some blue points that have to be visited by the drone in coordination with a base vehicle. The aim of the problem is to find some points where the drone is launched and retrieved from the mothership by taking into account that both vehicles are moving simultaneously and that the drone has a limited endurance to reach the blue points. The objective is to find a minimal cost route in terms of the distance made by the mothership satisfying all these requirements.

Here we can see the route made by the mothership, represented by the black line, and where the drone is launched and retrieved, represented by the red points.

In our problem, the Mothership and Drone Routing Problem with Graphs, we extended the case of visiting points to a more general structures given by graphs. Again, we have a mothership, as a base vehicle, and a drone, that has to visit the graphs, whose locations are given. For each graph, the drone performs the following task:

* It is launched from the current mothership location.
* It flies to the graph g that has to be visited.
* It traverses the required edges of the graph g.
* It returns to the current position of the mothership.

In summary, It is required to determine:

* The tour of the mothership starting at orig , deciding the different launching and rendezvous points, and returning to dest.
* The order of visits of the target graphs followed by the drone, determining the corresponding launching and rendezvous points of the drone on each visited graph.
* The tour followed by the drone on each target graph g ∈ G.

Depending on where the mothership is allowed to move, this problem produces two different variants:

* The case where the mothership can move freely on the continuous space, called the All-Terrain Mothership and Drone Routing Problem with Graphs (AMDRPG).
* The mothership can move on a road network (that is, it is a normal truck or van), called the Network Mothership-Drone Routing Problem with Graphs (NMDRPG). We are only focusing on the first variant, but the details of the other model can be found in the work.

Here, we show the parameters that our model has:

* Orig: coordinates of the point defining the origin of the mothership path.
* Dest: coordinates of the point defining the destination of the mothership path.
* G, representing the set of graphs.
* Lower case g, representing the nodes and edges of each target graph.
* L(eg), length of the Edge eg.
* Beg, Ceg, coordinates of the end points of the Edge eg.
* Alphaeg, percentage of the Edge eg that must be traversed.
* Alphag, percentage of the total length of the graph g that must be traversed.
* vD, vM, velocities of the drone and the mothership.

Here we can see a picture of the initial data of the problem, it is known the location of the graphs and the origin and destination.

Firstly, we formulated the problem by using an index t that represents the different 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.

Here, we present the binary variables of the model:

μeg ∈ {0, 1} ∀eg ∈ Eg (g ∈ G): equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, and 0 otherwise.

entry eg ∈ {0, 1} ∀eg ∈ Eg (g ∈ G): auxiliary binary variable used for linearizing expressions.

ueg t ∈ {0, 1} ∀eg ∈ Eg (g ∈ G) ∀t ∈ T: equal to 1 if the drone enters in graph g by eg at stage t, 0 otherwise.

zeg e′g ∈ {0, 1} ∀eg , e′g ∈ Eg (g ∈ G): equal to 1 if the drone goes from eg to e′g , 0 otherwise.

veg t ∈ {0, 1} ∀eg ∈ Eg (g ∈ G) ∀t ∈ T: equal to 1 if the drone exits from graph g by eg at stage t, 0 otherwise.

seg , ∀eg ∈ Eg (g ∈ G): integer non negative variable representing the order of visit of edge e.

Now, we show the location variables that says where the drone is launch and retrieved from the mothership, and the drone entry and exit points from each Edge of the graph.

Then, we also have the distance variables that model each movement that makes the drone and the mothership. These distances are modelled by using Second Order Cone constraints given in the following equations. The first four constraints model the distance covered by the drone whereas the last two take into account the distance made by the mothership.

We have considered two modes of visiting the graphs. We have modelled the case where the drone must traverse a given percentage of the length of the edge of the graph (called Alpha-e) or a percentage of the total length of the graph (called Alpha-g).

In this slide we show how to linearize this product of variables using the absolute value trick and the parameterization of the entry and exit points on the Edge e\_g, namely, rho\_eg and lambda\_eg.

We introduce the following equalities for the tour made by the drone, the first four assign one graph to one stage. The fifth equation links the u and v variables and the last two inequalities are flow conservation constraints applied to the edges of the graph.

In addition, it is necessary to prevent the existence of subtours inside the graphs. To avoid them, we can include the Miller-Tucker-Zemlin constraints or the exponential set of subtour elimination constraints.

Moreover, we have to include the coordination constraint, that ensures that the time spent by the drone to traverse the required percentage of the graph is less or equals to the one spent by the mothership to go from the launch point to the rendezvous point.

Finally, we set the origin and the destination point where the mothership and the drone must start and end, respectively.

And now, we show the complete formulation for the AMDRPG based on stages, where the three first addends address the weighted distance covered by the drone, while the last one is referred to the weighted distance made by the mothership. We assume that beta\_m is bigger than beta\_d, since the cost of using the mothership is also bigger.

Later, we noticed that we can associate each launch point and rendezvous point to each graph, avoiding the t index and stating other constraints to model the tour made by the mothership.

Here, we have to include the w variable, to decide what is the order of visiting the graphs.

Again, we have the location and distance variables and their respective inequalities, but the most significant difference is that the drone route constraints is much simpler and it is necessary to include some new constraints to the tour made by the mothership, among which are the MTZ or the subtour elimination constraints. Finally, we have the coordination constraint associated to each graph.

In the example 1, you can see some figures that represent the solution for each mode of visiting the graphs:

* The first picture shows the initial data.
* The second one is obtained by solving the AMDRPG restricted to visit the %50 of each Edge. The black line represents the tour made by the mothership, and the red line represents the movement made by the drone.
* The third picture is the optimal solution for the AMDRPG when we have to visit the %50 of the whole graph.

The example 2 shows how the solutions change when we change the required percentage of visiting each Edge of the graph and the example 3 is the analogous but when we require that we have to visit a percentage of the whole graph.

Since dealing with the exact model is a hard task for medium size instances, our strategy is to present our matheuristic approach to address the solution. The rationale of this algorithm rests on decomposing the problem in simpler subproblems decoupling the decisions made on the route followed by the mothership and the ones made on the drone.

Finally, we detail the computational experiments that we made by taking into account the different parameters that the problem has.

Firstly, we decided to have planar graphs. We designed grid graphs as you can see in the picture and we also used the scipy.spatil.Delauney Python class to generate delauney graphs, that are also planar.

In the experiment 1, we compare gurobi and cplex. We generated 5 instances of 10 delauney graphs, and 5 instances of 10 grid graphs with the following setting. The experiment consists on:

* Running the three formulations proposed for the (AMDRPG): Stages, MTZ and SEC.
* Using two commercial solvers, Cplex 12.8 and Gurobi 9.03.
* Time Limit: 1 hour.

The results show that Gurobi gap is better than the one provided by the cplex, and we also can see that stages formulation work worse than the MTZ and SEC. The results also say that the delauney instances are harder than the grid ones. Thats what we decided to use only the MTZ formulation with gurobi for the next experiment.

In the experiment 2, we generated 5 instances of 5, 10, 15, and 20 delauney and grid graphs, with the same setting, but requiring only the 20% of the graphs. The experiment consists on:

* Running the MTZ formulation for (AMDRPG) with and without initial solution provided by the matheuristic.
* Using Gurobi 9.03.
* Time Limit: 2 hours.

The first picture shows the matheuristic running time, that it increases with the number of targets to be visited both for Grid and Delaunay instances.

The second picture represents the 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.

The third picture shows the percentage gap of the solution provided by the exact solution of the MTZ formulation within the time limit without the initialization, with respect to the one found with the initialization.

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.