

DRONE ROUTING AND TRAJECTORY OPTIMISATION

Walton Pereira Coutinho

Universidade Federal de Campina Grande
Rua Luiz Grande, s/n, 58540-000, Sumé-PB
walton.coutinho@ufcg.edu.br

Jörg Fliege

University of Southampton
University Road, Highfield, Southampton, SO17 1BJ, United Kingdom
j.fliege@soton.ac.uk

Maria Battarra

University of Bath
Claverton Down, Bath, BA2 7AY, United Kingdom
m.battarra@bath.ac.uk

Anand Subramanian

Universidade Federal da Paraíba
R. dos Escoteiros, s/n, 58058-600, João Pessoa-PB
anand@ci.ufpb.br

ABSTRACT

In recent years, employing Unmanned Aerial Vehicles (UAVs), a.k.a. drones, to collect data has gained popularity. Often, the use of UAVs allows for a reduction in costs for many operations. However, constraints due to the flight dynamics have often been neglected. In this paper we introduce the Unmanned Aerial Vehicle Routing and Trajectory Optimisation Problem (UAVRTOP), the problem of optimising the routes and trajectories of a fleet of UAVs. Motivated by a disaster assessment application, we propose a variant in which a fleet of autonomous aerial gliders is due to photographing a set of waypoints. We propose a Mixed-Integer Non-linear Programming (MINLP) formulation for this variant. In addition, we present a heuristic method that is composed of a novel trajectory optimisation heuristic and a classical routing matheuristic. We perform computational experiments with real-life instances based on flood risk maps of cities in the UK.

KEYWORDS. Unmanned Aerial Vehicles. Drones. Routing. Trajectory Optimisation.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are aircraft that do not need a human pilot on board. In general, these vehicles are either controlled by an embedded computer or by a pilot operating a remote control. Drones, remote controlled helicopters and unmanned gliders are examples of UAVs. Gliders differ from the other types due to the lack of on-board propulsion.

The academic routing community has acknowledged the interest of companies and organisations in adopting UAVs in their operations. However, the UAV routing literature has often neglected constraints due to the flight dynamics of UAVs. For some real-world applications involving more complex UAV systems, such as unmanned gliders and fixed-wing vehicles, the definition of routes must be coupled to the design of flyable trajectories, otherwise the assigned routes might become inefficient or even infeasible for these UAVs.

Research about integrated routing and Trajectory Optimisation (TO) problems seems to be still fragmented. In this work, we introduce the UAV Routing and Trajectory Optimisation Problem (UAVRTOP), which consists of optimising the routes and trajectories for a fleet of UAVs subject to flight dynamics constraints. We believe that integrating TO and routing in a single optimisation problem is a key research challenge in adopting UAVs for real world applications.

In disaster assessment applications, UAVs play an important role on collecting information. The data generated by a UAV can be analysed in order to, e.g., coordinate rescue efforts, detect building failures, detect casualties in life-risk situations and allocate resources efficiently. An alternative to the use of expensive powered UAVs are low cost balloon-launched autonomous gliders [Crispin, 2016]. However, these gliders are highly dependent on wind conditions that are usually incorporated into flight dynamics. Therefore, integrating routing and trajectory optimisation is necessary for applications involving a fleet of gliders.

In this paper, we are interested on modelling and solving problems related to routing and trajectory optimisation of unmanned gliders, with applications on disaster assessment. We consider the situation where a fleet of gliders is required to fly over a set of waypoints subject to constraints on the flight dynamics. This problem will be referred to as the Glider Routing and Trajectory Optimisation Problem (GRTOP). The main motivation to solve this problem arises from aerial photographing missions in disaster situations, where a fleet of gliders is required to take pictures of a number of points of interest, e.g., hospitals, schools, residential areas and inaccessible areas, while optimising some performance criteria such as the mission time.

We propose a multi-phase Mixed-Integer Non-linear Programming (MINLP) formulation for the GRTOP in which the flight dynamics of the gliders are modelled as dynamic constraints. We do not attempt to solve this formulation directly, instead we propose a hybrid heuristic method that is composed of two main building blocks: (i) a Sequential Trajectory Optimisation (STO) heuristic, designed to cope with challenging task of finding feasible (flyable) trajectories for a given route; and (ii) a routing matheuristic, capable of generating routes that can be evaluated by the STO procedure, that combines iterated local search and a set-partitioning-based integer programming formulation. Computational experiments are performed on real-life instances based on flood risk maps of cities in the UK as well as in a large number of randomly generated instances.

The remainder of this paper is organised as follows. Section 2 reviews the recent contributions on routing and trajectory optimisation of UAVs. Section 3 presents the disaster assessment problem, the glider dynamics and a multi-phase MINLP formulation for the GRTOP. Section 4 introduces the proposed matheuristic. Section 5 shows the results of computational experiments. Finally, Section 6 concludes this work and provides future research directions.

2. Literature Review

In this section, we present recent contributions of articles that studied UAV routing and TO in an integrated framework.

Zhang et al. [2012] investigated the problem of routing a combat Unmanned Combat Aerial Vehicle (UCAV) in a 3D environment through stationary ground targets while avoiding no-fly threat areas. In order to succeed on the attacks, the UCAV must fly within the targets' allowable attack region (which consisted of a hollow-cone-like airspace around the target) and respect projectile release attitude and velocity constraints. The UCAV was modelled by high fidelity 3DOF Equations of Motion (EOMs) taking wind velocities into account.

In order to solve this problem, Zhang et al. [2012] propose a hierarchical heuristic with two levels. In the first level, the vehicle's state space is discretised into a set of feasible points that intersects the targets' allowable attack region by using a modified probabilistic road map method. Then, for every pair of sampled points not in the same target a TO problem was solved to obtain feasible trajectories (with respect to the vehicle's dynamics and operational constraints) and their respective costs. The second decision level consists of solving a Generalised Travelling Salesman

Problem (GTSP) over the network produced in the first level. This is accomplished by transforming the GTSP into an instance of the Asymmetric TSP by means of the noon-bean transformation method. The Lin-Kernighan heuristic was then employed to solve the ATSP. In addition, the authors embedded this algorithm into a real-time framework in order to make this approach more flexible in practical applications. Numerical experiments showed that this approach is computationally intensive. The authors reported that roughly 50 minutes were necessary to solve a test case with three targets and one no-fly zone.

Fügenschuh e Müllenstedt [2015] studied the problem of designing and routing a fleet of heterogeneous UAVs over a set of waypoints. The waypoints have to be selected from a list, where a score was associated to each waypoint. The objective was to maximise the total score (defined as the sum of the individual scores) while minimising the total flight time. The UAVs' motion was modelled by piecewise linear dynamics based on Newton's law of motion. The advantage of using this model lies on its simplicity, since the discretised version of these EOMs is also linear. On the other hand, the accuracy of such a model regarding UAVs flight dynamics is limited. In order to represent the range of the UAVs' sensors, the waypoints were considered to rest inside a sphere. A waypoint would be considered visited if a UAV passes through its covering sphere. No-fly zones and collision avoidance among the UAVs were also considered. Finally, different locations could be chosen to launch each UAV.

The authors proposed a MINLP formulation to this problem, which was linearised and could be solved by a commercial Mixed-Integer Linear Programming (MILP) optimisation software. Eight instances were created by varying the number of waypoints between 3 and 15, the number of no-fly zones between 0 and 3 and the number of UAVs between 1 and 2. Computational experiments showed that bigger instances with 10 and 15 waypoints could not be solved within one hour. The computation time required to solve smaller problems to optimality varied between 57 and 3400 seconds.

A similar approach was presented by Forsmo [2012]. The author applied Newton's second law in order to model the motion of the UAVs. However, constraints on the magnitudes of forces, velocities and yaw rates were also imposed, which increased the complexity of the physical representation of the UAVs. Several operational constraints were considered, such as obstacles and collision avoidance. Scenarios with up to two UAVs and multiple waypoints were generated. A MILP formulation was proposed in order to find minimum flight time trajectories visiting all waypoints subject to mission and operational constraints. Computational experiments were performed over 5 test cases, constructed by varying the number of UAVs (1 or 2), waypoints (6 or 8) and by imposing or not a visiting order. The authors showed that by CPU times could be reduced by decreasing the flight time horizon.

In Coutinho et al. [2018], the authors provide a formal definition of the UAV Routing and Trajectory Optimisation Problem (UAVRTOP). A general multi-phase mixed-integer optimal control formulation is proposed for this problem. Next, the authors introduce a taxonomy capable of classifying UAV routing TO problems according to their most relevant features. This taxonomy includes 20 attributes that are common in the literature. The literature on UAVs routing has been surveyed and a lack of articles integrating UAV routing and TO has been identified.

In Coutinho et al. [2019], the authors propose a single-phase Mixed-Integer Nonlinear Programming (MINLP) formulation for the GRTOP, which simultaneously optimises the routes as well as the flight path along these routes, while flight dynamics are modelled as constraints. They avoid solving a non-convex problem by linearising the gliders' flight dynamics, converting the proposed MINLP into a Mixed-Integer Second-order Cone Programming (MISOCP) problem. To allow for a more tractable formulation, the dynamical constraints are relaxed and a penalisation is added to the objective function. Several different discretisation techniques and solvers are compared.

3. An application in disaster assessment

We study a variant of the UAVRTOP where a fleet of balloon-lifted gliders is required to survey a number of points of interest, such as hospitals, schools and residential areas, in order to assess possible damage and people at risk in the aftermath of a disaster. Gliders are launched and are expected to land in one of the predetermined landing zones. The position of launch sites can be estimated using the tool “ASTRA High Altitude Balloon Flight Planner” proposed by Sobester et al. [2013], available at Zapponi [2013].

Each glider is equipped with a remote camera able to survey objects positioned within relative ranges. An inverted conic shape is adopted in order to model a cameras’ field of view (Figure 1). This type of geometry has also been used for UAV-camera systems in Ariyur e Fregene [2008], Roelofsen et al. [2016] and Nedjati et al. [2016]. We assume that cameras are fixed to the body of the gliders, and we enforce the gliders to fly in level-flight (or “flat”) over a waypoint in order to properly photograph the desired object. For the sake of simplicity, we assume that the fleet of gliders is homogeneous and cameras have the same specifications. For each waypoint, the cameras’ field of view allows us to define conic-like regions that must be visited by the gliders.

Figure 2 illustrates the geometric representation of a waypoint. In this picture, the object of interest corresponds to the blue box. Each waypoint is entirely described by $(\bar{x}_i, \bar{y}_i, \bar{r}_i, \bar{h}_i, \bar{h}_i), i \in V$, where $(\bar{x}_i, \bar{y}_i)$ represents the position of the object i in the xy plane. Parameter $\bar{r}_i > 0$ represents the radius of a circle in the xy plane enclosing the footprint of the object i . Parameters $\bar{h}_i$ and $\bar{h}_i$ denote the minimum and maximum heights in which object i can be photographed, respectively. The last two components define and constrain the quality of the pictures. Without loss of generality, we set the cameras’ opening angle α to 45° while the points of interest are assumed to lie in the same xy plane (i.e., the altitude of each waypoint $i \in V$ is assumed to be 0). Considering the aforementioned assumptions, a glider flying at an altitude h can visit a waypoint i and take a good picture if it touches or enters the truncated cone covering i , i.e., if $(x - \bar{x}_i)^2 + (y - \bar{y}_i)^2 \leq (h + \bar{r}_i)^2 \tan^2 \alpha = (h + \bar{r}_i)^2$.

Landing zones can be defined in a similar way. The tuple $(\tilde{x}_i, \tilde{y}_i, \tilde{r}_i)$ describes the geometry of a landing zone $i \in L$. The first two components define position on the xy plane and $\tilde{r}_i$ the radius of the landing site. Without loss of generality, we will assume that $\tilde{h}_i$ equals 0 for all $i \in L$. The shape of landing zones consists of half-spheres with centres in the xy plane.

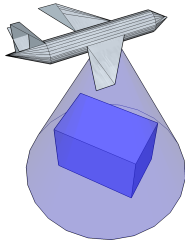


Figure 1: Camera model.

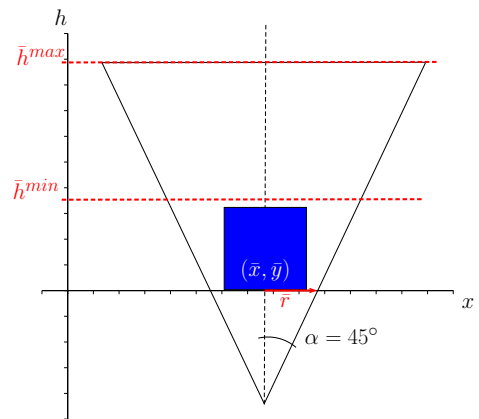


Figure 2: Waypoint geometry.

3.1. Glider flight dynamics

The motion of an aerial glider can be modelled by a set of Ordinary Differential Equations (ODEs). One popular approach consists of employing a set of three-dimensional kinematics and dynamics EOMs for rigid bodies, such as the ones presented by Zhao [2004].

Let us define the *state* of a glider at time $\tau \in \mathbb{R}_{\geq 0}$ as a *state vector* $\mathbf{y}(\tau) = (x(\tau), y(\tau), h(\tau), v(\tau), \gamma(\tau), \varphi(\tau))^T$, where the first three components $x(\tau), y(\tau)$ and $h(\tau)$ of $\mathbf{y}(\tau)$ are the position of the glider and $v(\tau) \in \mathbb{R}_{\geq 0}$ is the relative velocity (a.k.a. *airspeed*) of the aircraft with respect to the wind velocity. Finally, $\gamma(\tau) \in \mathbb{R}$ is the air-relative flight path angle and $\varphi(\tau) \in \mathbb{R}$ is the heading angle. The *controls* (or input) of the system are represented by the *control vector* $\mathbf{u}(\tau) = (C_L(\tau), \mu(\tau))^T$, where $C_L(\tau) \in \mathbb{R}$ is the lift coefficient and $\mu(\tau) \in \mathbb{R}$ the bank angle. In the following, the notation “ $\dot{\cdot}$ ” is used to represent time derivatives of the variables.

The EOMss of a glider can be expressed by Equations (1–6). For the sake of notation simplicity, we omit the dependence on time (τ) from state, control and auxiliary variables.

$$m_g \dot{v} = -D - m_g g_e \sin \gamma - m_g \dot{U} \cos \gamma \sin \varphi \quad (1)$$

$$m_g v \dot{\gamma} = L \cos \mu - m_g g_e \cos \gamma + m_g \dot{U} \sin \gamma \sin \varphi \quad (2)$$

$$m_g v \cos \gamma \dot{\varphi} = L \sin \mu - m_g \dot{U} \cos \varphi \quad (3)$$

$$\dot{x} = v \cos \gamma \sin \varphi + U(h) \quad (4)$$

$$\dot{y} = v \cos \gamma \cos \varphi \quad (5)$$

$$\dot{h} = v \sin \gamma, \quad (6)$$

In Equations (1–6), the wind’s velocity is $U(h)$. The auxiliary variables D and L represent the drag and lift forces, respectively, acting on the glider. The definition of lift and drag equations as well as the remaining auxiliary variables can be found in [Betts, 2001].

By re-writing Equations (1–6) so that the time derivatives are isolated and by grouping the equations, one can obtain a compact representation of the system dynamics as follows:

$$\dot{\mathbf{y}} = f(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau), \quad (7)$$

where $f(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau)$ corresponds to the right-hand-side of the system of ODEs.

3.2. Linearisation and discretisation of the glider’s dynamics

Embedding Equations (1–6) into an optimisation problem generally leads to non-convex formulations. Local optimisation software for non-linear optimisation often requires high quality initial guesses. In order to avoid this issue, we linearise the gliders’ EOMss. This simplification is usually preferred in the literature when the dynamics are very non-linear [Ahmed et al., 2015; Hajiyev et al., 2015; How et al., 2015].

The glider’s equations of motion have been linearised by applying the first order Taylor’s expansion around certain equilibrium conditions. These equilibrium conditions were defined under the assumption that all forces acting upon the glider are summing up to zero. After applying Taylor, we define a linear system of differential equations that describe the movement of the glider.

In this work, we solve the gliders’ EOMss by means of a *direct collocation* method [Betts, 2001]. This is accomplished by discretising the linear EOMss and adding the resulting expressions as constraints in the GRTOP. Since the flight time interval $[\tau_o, \tau_f]$ and step size η will be fixed for all methods presented here, the linearity of the EOMss is maintained.

After solving a numerical integration procedure, it is necessary to approximate the actual solution between collocation points by using a piece-wise polynomial function (e.g., a spline). Usually, the degree of the polynomial function for approximating the solution is chosen according to the order of accuracy of the integration method. For example, if a second order method is chosen in the integration step, then a quadratic spline should be applied to interpolate the controls and a third order spline should approximate the states.

Euler’s discretisation, for example, works by replacing the controls and system dynamics by piece-wise linear approximations. Therefore, after solving the numerical integration, one needs to reconstruct the control and system dynamics trajectories by using a linear interpolation. Let us

define the independent variable τ in terms of subsequent time steps t and $t + 1$, i.e., $\tau \in [\tau_t, \tau_{t+1}]$. An approximation for the control function can be written as in Equation (8).

$$\mathbf{u}(\tau) = \mathbf{u}_t + \frac{\tau - \tau_t}{\eta}(\mathbf{u}_{t+1} - \mathbf{u}_t) \quad (8)$$

The approximation for the system dynamics also follows a linear function (Equation (9)). Moreover, by integrating both sides of this approximation, as in Equation (10), one finds a quadratic interpolation for the state function, expressed by Equation (11). An appropriate choice for the integration constant is $c = \mathbf{y}_t$.

$$\frac{d\mathbf{y}}{d\tau} = \mathbf{f}_t + \frac{\tau - \tau_t}{\eta}(\mathbf{f}_{t+1} - \mathbf{f}_t) \quad (9)$$

$$\int d\mathbf{y} = \int \left(\mathbf{f}_t + \frac{\tau - \tau_t}{\eta}(\mathbf{f}_{t+1} - \mathbf{f}_t) \right) d\tau \quad (10)$$

$$\mathbf{y}(\tau) = \mathbf{f}_t(\tau - \tau_t) + \frac{(\tau - \tau_t)^2}{2\eta}(\mathbf{f}_{t+1} - \mathbf{f}_t) + c \quad (11)$$

3.3. Multi-phase MINLP formulation

We associate to each arc $(i, j) \in A$ a phase of a glider's trajectory. In this paper, we use the words phase and arc interchangeably. Let us define $\mathbf{y}_{ijg}(\tau_{ijg}) : [0, \infty) \rightarrow \mathbb{R}^6$ and $\mathbf{u}_{ijg}(\tau_{ijg}) : [0, \infty) \rightarrow \mathbb{R}^2$ as the state and control variables, respectively, of glider $g \in G$ flying through arc $(i, j) \in A$ at time $\tau_{ijg} \in \mathbb{R}$. Let variables τ_{ijg}^o and τ_{ijg}^f represent the initial and final flight times of the glider g flying on arc (i, j) such that $\tau_{ijg} \in [\tau_{ijg}^o, \tau_{ijg}^f]$.

Let us define variable $a_{ijg} \in \{0, 1\}$ that takes value 1 if glider g traverses arc $(i, j) \in A$ and taking value 0 otherwise. For simplicity, we define the set A' as the set of arcs not leaving from the launching point, i.e., $A' = A \setminus \{(0, j), j \in V\}$. The UAVRTOP can be formulated as:

$$\min \max_{g \in G} \left\{ \sum_{(i,j) \in A} (\tau_{ijg}^f - \tau_{ijg}^o) a_{ijg} \right\} \quad (12)$$

$$\sum_{i \in V} a_{0ig} = \sum_{i \in V} a_{ilg} \leq 1, \forall l \in L, \forall g \in G \quad (13)$$

$$\sum_{g \in G} \sum_{i \in V} a_{ijg} \geq 1, \forall j \in V, j \neq i \quad (14)$$

$$\sum_{i \in V} a_{ijg} - \sum_{i \in V} a_{jig} = 0, \forall j \in V, j \neq i, \forall g \in G \quad (15)$$

$$a_{ijg}((x_{ijg}(\tau_{ijg}^o) - \bar{x}_i)^2 + (y_{ijg}(\tau_{ijg}^o) - \bar{y}_i)^2) \leq (h_{ijg}(\tau_{ijg}^o) + \bar{r}_i)^2, \forall (i, j) \in A', \forall g \in G \quad (16)$$

$$a_{ijg} \underline{h}_i \leq a_{ijg} h_{ijg}(\tau_{ijg}^o) \leq \bar{h}_i, \forall (i, j) \in A', \forall g \in G \quad (17)$$

$$a_{ijg}((x_{ijg}(\tau_{ijg}^f) - \bar{x}_i)^2 + (y_{ijg}(\tau_{ijg}^f) - \bar{y}_i)^2 + h_{ijg}^2(\tau_{ijg}^f)) \leq \bar{r}_j^2, \forall i \in V, \forall j \in L, \forall g \in G \quad (18)$$

$$-\hat{\gamma} \leq a_{ijg} \gamma_{ijg}(\tau_{ijg}^o) \leq \hat{\gamma}, \forall (i, j) \in A', \forall g \in G \quad (19)$$

$$-\hat{\mu} \leq a_{ijg}\mu_{ijg}(\tau_{ijg}^o) \leq \hat{\mu}, \forall (i, j) \in A', \forall g \in G \quad (20)$$

$$\dot{\mathbf{y}}_{ijg} = \mathbf{f}_g(\mathbf{y}_{ijg}(\tau_{ijg}), \mathbf{u}_{ijg}(\tau_{ijg}), \tau_{ijg})a_{ijg}, \forall (i, j) \in A, \forall g \in G \quad (21)$$

$$\mathbf{y}_{jkg}(\tau_{jkg}^o) = \mathbf{y}_{ijg}(\tau_{ijg}^f)a_{ijg}a_{jkg}, \forall (i, j), (j, k) \in A, \forall g \in G \quad (22)$$

$$\mathbf{u}_{jkg}(\tau_{jkg}^o) = \mathbf{u}_{ijg}(\tau_{ijg}^f)a_{ijg}a_{jkg}, \forall (i, j), (j, k) \in A, \forall g \in G \quad (23)$$

$$\tau_{jkg}^o = \tau_{ijg}^f a_{ijg}a_{jkg}, \forall (i, j), (j, k) \in A, \forall g \in G \quad (24)$$

$$\tau_{ijg}^o \leq \tau_{ijg} \leq \tau_{ijg}^f, \forall (i, j) \in A, \forall g \in G \quad (25)$$

$$a_{ijg} \in \{0, 1\}, \forall (i, j) \in A, \forall g \in G \quad (26)$$

$$\mathbf{y}_{ijg}(\tau_{ijg}) \in \mathbb{R}^6, \forall (i, j) \in A, \forall g \in G \quad (27)$$

$$\mathbf{u}_{ijg}(\tau_{ijg}) \in \mathbb{R}^2, \forall (i, j) \in A, \forall g \in G \quad (28)$$

$$\tau_{ijg}^o, \tau_{ijg}^f \in \mathbb{R}, \forall (i, j) \in A, \forall g \in G. \quad (29)$$

In this formulation, the objective function (12) minimises the flight duration of the longest route. Constraints (13–15) account for the assignment of routes to gliders. Constraint (13) ensures that every used glider lands in one of the predetermined landing sites. Constraints (14) and (15) make sure that every waypoint is visited at least once and that the continuity of routes is preserved. Constraints (16–18) ensure that the gliders lie within the respective covering regions of the waypoints and landing sites at the end points of each phase. Constraints (16) and (17) ensure that a glider g flies within the boundaries of waypoint i at time τ_{ijg}^o , if arc (i, j) is used, while Constraints (18) state that a glider g must lie within the boundaries of a landing site at the end of its mission. Constraints (19) and (20) make sure that glider g is nearly in a “flat” position (i.e., in level flight within very small pitch ($\hat{\gamma}$) and roll ($\hat{\mu}$) angles) at the moment it takes a picture of waypoint i . Constraints (21–25) are related to the flight dynamics of the gliders. Constraints (21) enforce the flight dynamics of glider g to be applied if this glider flies through arc (i, j) . Constraints (22–24) ensure the continuity of state, control and time variables is maintained if arc (i, j) precedes arc (j, k) in a solution. Constraints (25) preserve the time variable within its bounds on arc (i, j) . Finally, Constraints (26–29) define the domain of the optimisation variables.

4. A heuristic routing and trajectory optimisation method

This section describes the proposed matheuristic algorithm, denoted as ILS-STO, for solving the GRTOP. Sequential Trajectory Optimisation (STO) is a heuristic developed to find feasible trajectories for a glider flying through several waypoints. The developed approach basically consists of a multi-start Iterated Local Search (ILS) combined with a Set Partitioning (SP) integer programming formulation, and STO.

The matheuristic procedure performs I_R restarts where at each of them an initial solution is generated using a greedy randomised insertion heuristic, whose detail description can be found in Subramanian [2012]. Since we aim at minimising the maximum route cost, the only routes considered for insertion in a particular iteration are those where its associated estimate cost is strictly smaller than the estimate maximum route cost of the partial solution. The method then iteratively tries, for I_{ILS} iterations, to improve the initial solution by means of local search and perturbation mechanisms. If a solution s is improved (with respect to $f(\cdot)$) after the local search phase, then the best current solution is updated. The pool of routes (R_{pool}) is updated by adding the feasible routes from s . The best solution found after each restart is possibly updated in lines. Finally, the algorithm tries to find the optimal combination of the feasible routes stored in R_{pool} by solving a SP-based problem using a general purpose MILP solver. Algorithm 1 shows the pseudocode of ILS-STO.

Our STO heuristic can be described as follows. Let S be the set of all subsequences (or routes) of V starting at 0 and ending at a launching point in L , such that every waypoint is visited exactly once. W.l.o.g., we can define an element r of S as $r = (i_0, i_1, \dots, i_l), i_1, \dots, i_{l-1} \in V$ and $i_l \in L$. The trajectory of a glider flying through route r is then divided into $n_r - 1$ phases

Algorithm 1 ILS-STO

```
1: Procedure ILS-STO ( $I_R, I_{ILS}, FT_{Max}$ )
2:  $f^* \leftarrow \infty$ ;  $Rpool \leftarrow \text{NULL}$ 
3: for  $i=1, \dots, I_R$  do
4:    $s \leftarrow \text{GenerateInitialSolution}()$ 
5:    $s' \leftarrow$  artificial solution with very large flight cost ( $f(s') \leftarrow \infty$ )
6:    $iter \leftarrow 0$ 
7:   while  $iter \leq I_{ILS}$  do
8:      $s \leftarrow \text{LocalSearch}(s)$ 
9:      $f(s) \leftarrow \max_{r \in s} \{f(r) \leftarrow \text{STO}(r, FT_{Max})\}$ 
10:    if  $f(s) < f(s')$  then
11:       $s' \leftarrow s$ 
12:       $iter \leftarrow 0$ 
13:     $Rpool \leftarrow Rpool \cup$  feasible routes from  $s$ 
14:     $s \leftarrow \text{Perturb}(s')$ 
15:     $iter \leftarrow iter + 1$ 
16:    if  $f(s') < f^*$  then
17:       $s^* \leftarrow s'$ ;  $f^* \leftarrow f(s')$ 
18:   $s^* \leftarrow \text{SP}(s^*, Rpool)$ 
19: return  $s^*$ 
20: end ILS-STO.
```

according to each arc of the route, where n_r denotes the number of elements in route r . For a fixed flight time, each single-phase TO between two waypoints can be solved to optimality as a Second-Order Cone Programming (SOCP). The minimum flight time on each phase is computed heuristically by an incremental method. Each phase in route r is then connected in such a way that the initial conditions of arc (j, k) equals the final conditions of arc (i, j) if waypoints i, j and k are present in route r in this specific order.

5. Computational experiments

The computational experiments described in the next sections were executed in an Intel Core i7-4770 CPU with 3.40GHz and 16GB of RAM running under Linux Mint 17 64bits (kernel 3.13.0-24). ILS-STO was coded in C++ and CPLEX 12.7 was used, in its standard configuration, to solve the SOCP subproblems.

In this section, the method for generating instances closely follows the procedure presented in Coutinho et al. [2019]. We have generated instances having $n \in \{10, \dots, 50\}$ waypoints and $m \in \{3, 4, 5\}$ landing zones. The number of gliders is computed by $n_g = \lfloor n/2 \rfloor, n_g \leq 10$. Five different instances were created for each combination of number of waypoints and landing zones. Let $\mathcal{U}[a, b]$ denote the continuous uniform distribution from a to b . The launching altitude h_o , in km , has been chosen from $\mathcal{U}[4, 5]$. These limits on the launching altitude are based on the values used by Crispin [2016] for an area of $10km^2$.

5.1. Results on real-life based flood scenarios

A number of instances based on UK cities prone to flooding has been created. They represent hypothetical flooding scenarios in the cities of Boston, Highbridge, London, Moore (Warrington) and Portsmouth. The so-called *UK instances* have been constructed as follows. We searched for flood risk-prone cities in the UK by using risk information maps provided at Government Digital Service [2017].

For each city, waypoints have been placed in locations with a high concentration of potential flooding victims, such as hospitals, schools, nurseries, residential areas, asylums and industries (which could become possible environmental hazards in a disaster situation), using Google maps. Next, the geographic coordinates of waypoints were collected and converted into Euclidean coordinates. The geometry of the waypoints and landing sites has been chosen in order to match

the dimensions of the real locations. Figures 3 and 4 depict the selection of waypoints for two flood-prone areas of two cities in the UK, namely, London and Highbridge. Figure 5 and 6 depict the solutions for 2 UK instances, namely, grtopS_lond1 (London) and grtopS_highb (Highbridge).

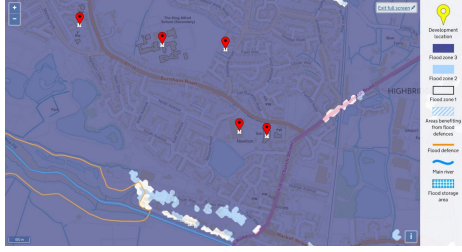


Figure 3: Waypoints selection for grtopS_lond1.

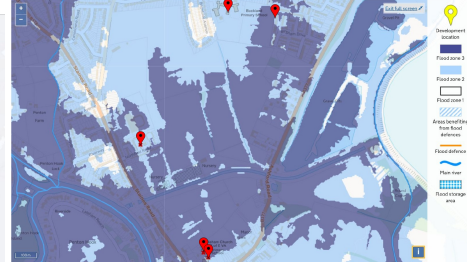


Figure 4: Waypoints selection for grtopS_highb.

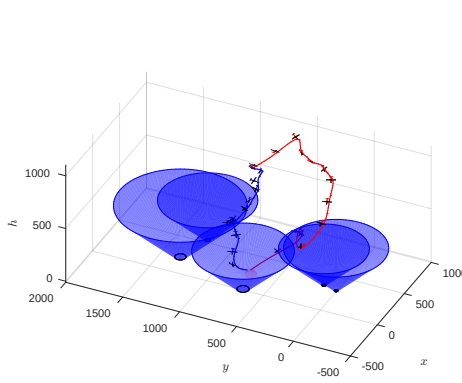


Figure 5: Solution of grtopS_lond1.

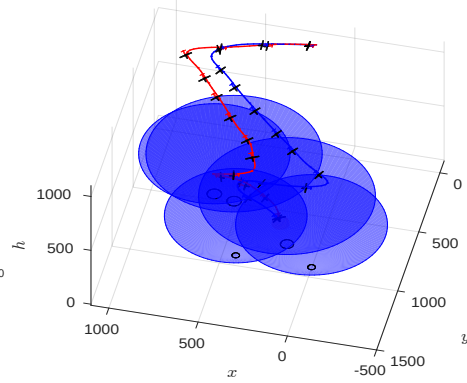


Figure 6: Solution of grtopS_highb.

5.2. Results of the ILS-STO heuristic

We ran ILS-STO ten times for each instance and the results obtained are reported in Table 1. In this table, **Instance** denotes the name of the instance. **Best Obj.** and **Avg. Obj.** correspond to the best and average solution costs, respectively. **Gap (%)** is the average gap between the average and the best solution costs computed as $100 \times [(Avg. Obj. - Best Obj.) / Best Obj.]$. **Time(s)** represents the average makespan and **Error** the average value of ϵ . **CPU(s)** is the average CPU time in seconds spent by ILS-STO, whereas **TO(%s)** is the proportion of **CPU(s)** spent on STO alone.

Table 1: Computational results for the randomly generated instances.

Instance	Best Obj.	Avg. Obj.	Gap(%)	Time(s)	Error	CPU(s)	TO(%s)
grtopL_103_1	674.32	674.32	0.00	561.38	112.95	27.42	99.89
grtopL_103_2	373.51	373.51	0.00	311.36	62.15	35.90	99.89
grtopL_103_3	538.21	538.21	0.00	501.55	36.66	44.52	99.89
grtopL_103_4	398.73	399.00	0.00	335.93	62.79	43.15	99.90
grtopL_103_5	524.90	524.89	0.00	455.51	69.38	29.35	99.89
grtopL_104_1	369.78	414.52	12.10	341.66	72.85	38.36	99.89
grtopL_104_2	416.04	418.81	0.33	375.49	41.94	37.24	99.89
grtopL_104_3	481.18	481.39	0.02	434.64	46.64	28.31	99.90
grtopL_104_4	390.38	443.45	13.42	393.14	49.64	36.04	99.90
grtopL_104_5	422.27	422.27	0.00	360.92	61.35	45.63	99.89
grtopL_105_1	469.63	469.63	0.00	433.88	35.75	40.21	99.88
grtopL_105_2	355.34	373.46	3.31	312.54	54.58	44.93	99.89

continues in the next page

Table 1 – continued from previous page

Instance	Best Obj.	Avg. Obj.	Gap(%)	Time(s)	Error	CPU(s)	TO(%)
grtopL_105_3	346.34	346.34	0.00	311.24	35.10	31.20	99.90
grtopL_105_4	365.56	389.14	4.63	310.16	72.32	33.62	99.89
grtopL_105_5	400.67	416.75	2.15	360.57	48.72	37.00	99.89
grtopL_203_1	344.66	349.92	0.00	303.62	41.04	92.17	99.88
grtopL_203_2	453.83	453.83	0.00	404.72	49.11	79.76	99.89
grtopL_203_3	411.97	413.03	0.26	374.58	38.45	80.51	99.89
grtopL_203_4	456.31	456.62	0.06	398.38	58.23	60.86	99.90
grtopL_203_5	454.26	454.26	0.00	395.70	58.56	71.18	99.90
grtopL_204_1	402.10	412.64	1.39	368.30	39.40	69.42	99.87
grtopL_204_2	444.87	475.13	6.37	389.15	84.06	62.66	99.89
grtopL_204_3	543.11	543.11	0.00	472.21	70.91	59.25	99.87
grtopL_204_4	480.02	480.46	0.03	446.92	33.26	89.72	99.89
grtopL_204_5	354.60	362.92	0.53	303.35	53.11	105.62	99.88
grtopL_205_1	354.50	361.31	1.54	306.10	53.86	72.99	99.88
grtopL_205_2	428.15	429.45	0.25	385.61	43.62	80.54	99.90
grtopL_205_3	440.20	440.20	0.00	396.93	43.27	78.76	99.90
grtopL_205_4	396.84	396.84	0.00	356.98	39.85	79.49	99.89
grtopL_205_5	396.63	396.63	0.00	326.59	70.04	68.98	99.88
grtopL_303_1	588.31	596.05	0.92	522.61	71.12	126.54	99.86
grtopL_303_2	503.17	507.25	0.81	444.49	62.77	96.35	99.89
grtopL_303_3	389.54	393.47	0.00	360.83	28.71	129.48	99.85
grtopL_303_4	425.56	436.05	1.81	384.10	49.15	146.09	99.87
grtopL_303_5	415.57	438.65	5.07	348.30	88.32	117.28	99.84
grtopL_304_1	400.11	406.88	0.37	359.96	41.64	133.73	99.84
grtopL_304_2	375.13	381.04	0.45	327.71	49.09	138.75	99.85
grtopL_304_3	374.75	377.88	0.00	341.29	33.46	135.53	99.86
grtopL_304_4	378.89	404.03	2.93	354.79	35.21	116.23	99.84
grtopL_304_5	421.70	422.31	0.15	366.61	55.71	127.02	99.86
grtopL_305_1	504.22	504.22	0.00	465.32	38.90	132.55	99.87
grtopL_305_2	411.22	416.08	0.00	388.40	22.82	172.39	99.87
grtopL_305_3	444.09	444.09	0.00	396.10	47.99	117.51	99.85
grtopL_305_4	358.85	366.87	1.41	314.59	49.32	121.17	99.84
grtopL_305_5	395.76	411.80	3.43	336.88	72.44	93.03	99.85
grtopL_403_1	438.00	449.17	1.18	392.04	51.11	179.79	99.82
grtopL_403_2	483.42	483.69	0.05	431.97	51.69	120.92	99.86
grtopL_403_3	409.65	409.65	0.00	348.33	61.32	135.14	99.79
grtopL_403_4	472.97	487.56	1.01	410.84	66.91	184.69	99.82
grtopL_403_5	390.15	408.51	2.01	364.11	33.89	194.53	99.82
grtopL_404_1	443.04	443.86	0.14	408.25	35.42	174.97	99.84
grtopL_404_2	474.87	474.87	0.00	415.63	59.23	145.55	99.82
grtopL_404_3	379.25	388.37	1.37	332.28	52.15	179.84	99.79
grtopL_404_4	479.75	488.80	0.32	440.75	40.53	173.03	99.82
grtopL_404_5	510.66	523.30	1.34	472.18	45.33	153.21	99.85
grtopL_405_1	511.35	526.68	3.00	446.68	80.00	135.54	99.85
grtopL_405_2	389.05	397.44	0.92	349.57	43.07	187.04	99.80
grtopL_405_3	424.95	427.46	0.07	387.33	37.90	197.51	99.81
grtopL_405_4	462.53	483.55	1.30	421.97	46.56	172.52	99.83
grtopL_405_5	450.86	456.31	0.34	398.53	53.84	177.90	99.83
grtopL_503_1	417.98	425.57	0.29	372.67	46.51	251.62	99.78
grtopL_503_2	381.15	393.29	1.54	343.96	43.06	232.28	99.76
grtopL_503_3	447.50	449.18	0.09	410.16	37.74	238.63	99.83
grtopL_503_4	487.47	494.21	1.35	444.66	49.39	159.25	99.84
grtopL_503_5	409.65	431.03	2.22	368.68	50.08	244.63	99.76
grtopL_504_1	398.75	400.93	0.12	358.41	40.80	260.52	99.79
grtopL_504_2	433.49	448.69	1.19	383.27	55.39	219.62	99.81
grtopL_504_3	464.63	473.04	1.81	415.38	57.66	226.94	99.79
grtopL_504_4	414.33	428.03	0.93	380.53	37.66	230.23	99.79
grtopL_504_5	451.14	460.33	1.08	413.30	42.73	286.28	99.82
grtopL_505_1	371.25	388.35	3.61	335.19	49.47	254.46	99.79
grtopL_505_2	424.97	434.93	1.14	381.41	48.38	229.34	99.79
grtopL_505_3	450.11	455.51	0.40	412.01	39.90	250.38	99.81
grtopL_505_4	354.30	380.03	2.82	321.06	43.24	269.68	99.77
grtopL_505_5	412.76	418.05	0.80	365.42	50.65	278.35	99.77

From the results reported on Table 1, we can observe that it takes, on average, 385 seconds to photograph waypoints distributed in an area of $10km^2$. The minimum amount of time necessary to finish a mission was around 300 seconds, while the maximum was around 560 seconds. It is

important to notice that the makespan does not seem to strongly depend on the number of waypoints of an instance. Finally, the error associated to each solution takes a value of $\varepsilon = 51.17$, on average. In Section 5, we showed that the value of ε is mostly associated to the value of the position variables (x, y, h) , which may take values of the order of km in our instances. Therefore, the average value of the error can be considered negligible when compared to the magnitude of the position variables. In fact, the maximum value of the error term in our experiments was 122.95, which is also negligible in comparison with the magnitudes of x, y and h .

From a computational point of view, one can observe that ILS-STO is capable of producing feasible solutions with a small computational effort. The average CPU time spent by the algorithm was never larger than 287 seconds, while the overall average CPU time to solve an instance was 130 seconds. We point out that STO is clearly the bottleneck of the method, as it substantially requires more CPU time than the other steps (as shown in column TO(%s)). From the relatively small values of the average gaps, we can verify that the solutions achieved are rather consistent. The overall average gap was 1.28%. Only in three cases the average gap was larger than 5%. For grtopL104_1 and grtopL104_4, the average gap was larger than 12%.

6. Conclusions

In this paper, we presented a heuristic algorithm for the GRTOP. We model the TO as a multi-phase optimal control problem, where the dynamics is allowed to change within a given route. We proposed a matheuristic algorithm for finding feasible solutions with a small computational effort. The so-called ILS-STO heuristic is composed of two main building blocks: (i) a Sequential Trajectory Optimisation (STO) heuristic to find feasible trajectories for a fixed sequence; and (ii) a routing ILS-SP matheuristic, which combines iterated local search and a set-partitioning formulation, for finding sequences of waypoints that can be evaluated by STO.

Our STO heuristic simplifies the multi-phase trajectory optimisation into a sequence of single-phase subproblems, which are formulated as SOCPs. Next, each subproblem is solved according to the sequence defined by the provided route. A feasible trajectory is then constructed by patching the solutions of each subproblem together in the provided order. Routes are generated by an ILS-based framework. Our routing matheuristic is based on the algorithm proposed by Subramanian [2012], where a random variable neighbourhood search is combined with a ILS metaheuristic and a SP formulation.

We tested the proposed algorithm on 75 new randomly generated instances. Results from our computational experiments showed that the proposed method is capable of finding feasible solutions in short computing times. The time required to find a local optimal solution was never superior to 290 seconds while the overall average gap was only around 1.28%, which shows that our method is sufficiently robust. Our experiments also showed that the SP-based formulation often helps on improving the objective function value. Our second set of experiments studied the effect of varying the maximum allowed number of gliders in the fleet for a subset of instances. Results showed that a fleet containing between 7 and 10 gliders usually produces better solutions in terms of makespan and fleet utilisation.

Future research avenues involving the GRTOP could consider alternative objective functions, such as optimising the fleet size and mission costs. Alternatively, equity objectives (e.g., optimising latency or the latest arrival) could also be taken into account. Moreover, the launching position of the gliders can as well be optimised as it might affect service times. Finally, by employing more accurate glider's dynamics one could achieve more realistic solutions. Nonetheless, this would require more sophisticated trajectory optimisation methods.

References

Ahmed, E., Hafez, A., Ouda, A., Ahmed, H., e Abd-Elkader, H. (2015). Modelling of a Small Unmanned Aerial Vehicle. *Advances in Robotics & Automation*, 4(126).

- Ariyur, K. B. e Fregene, K. O. (2008). Autonomous tracking of a ground vehicle by a UAV. In *2008 American Control Conference*, p. 669–671. IEEE.
- Betts, J. T. (2001). *Practical methods for optimal control using nonlinear programming*. Advances in design and control. Society for Industrial and Applied Mathematics, Philadelphia, PA. ISBN 978-0-89871-488-3.
- Coutinho, W. P., Battarra, M., e Fliege, J. (2018). The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review. *Computers & Industrial Engineering*, 120:116 – 128.
- Coutinho, W. P., Fliege, J., e Battarra, M. (2019). Glider routing and trajectory optimisation in disaster assessment. *European Journal of Operational Research*, 274:1138–1154.
- Crispin, C. (2016). *Path Planning Algorithms for Atmospheric Science Applications of Autonomous Aircraft Systems*. Ph.D. Thesis, University of Southampton, Southampton, UK.
- Forsmo, E. J. (2012). *Optimal Path Planning for Unmanned Aerial Systems*. Msc. Thesis, Norwegian University of Science and Technology, Norway.
- Fügenschuh, A. e Müllenstedt, D. (2015). Flight Planning for Unmanned Aerial Vehicles. Technical Report AMOS #34(2015), Helmut Schmidt University / University of the Federal Armed Forces Hamburg. Available at https://www.hsu-hh.de/download-1.5.1.php?brick_id=fyCYI55SQ6oFqe5e.
- Government Digital Service (2017). Long term flood risk information. <https://flood-warning-information.service.gov.uk/long-term-flood-risk/map?> Accessed: 20/05/2017.
- Hajiyev, C., Soken, H. E., e Vural, S. Y. (2015). Equations of Motion for an Unmanned Aerial Vehicle. In *State Estimation and Control for Low-cost Unmanned Aerial Vehicles*, p. 9–23. Springer International Publishing. ISBN 978-3-319-16416-8 978-3-319-16417-5.
- How, J. P., Frazzoli, E., e Chowdhary, G. V. (2015). Linear Flight Control Techniques for Unmanned Aerial Vehicles. In Valavanis, K. P. e Vachtsevanos, G. J., editors, *Handbook of Unmanned Aerial Vehicles*, p. 529–576. Springer Netherlands. ISBN 978-90-481-9706-4 978-90-481-9707-1.
- Nedjati, A., Izbirak, G., Vizvari, B., e Arkat, J. (2016). Complete coverage path planning for a multi-uav response system in post-earthquake assessment. *Robotics*, 5(4). ISSN 2218-6581. URL <http://www.mdpi.com/2218-6581/5/4/26>.
- Roelofsen, S., Martinoli, A., e Gillet, D. (2016). 3d collision avoidance algorithm for Unmanned Aerial Vehicles with limited field of view constraints. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, p. 2555–2560.
- Sobester, A., Castro, I. P., Czerski, H., e Zapponi, N. (2013). Notes on Meteorological Balloon Mission Planning. In *AIAA Balloon Systems (BAL) Conference*, p. 1–15. American Institute of Aeronautics and Astronautics.
- Subramanian, A. (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. Ph.D. Thesis, Universidade Federal Fluminense, Niterói.
- Zapponi, N. (2013). Astra high altitude balloon flight planner. <http://astra-planner.soton.ac.uk/>. Accessed: 21/02/2017.

- Zhang, Y., Chen, J., e Shen, L. (2012). Hybrid hierarchical trajectory planning for a fixed-wing UCAV performing air-to-surface multi-target attack. *Journal of Systems Engineering and Electronics*, 23(4):536–552.
- Zhao, Y. J. (2004). Optimal patterns of glider dynamic soaring. *Optimal Control Applications and Methods*, 25(2):67–89.